

21.658/IA/H/05



MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

PERANCANGAN DAN PEMBUATAN
PERANGKAT LUNAK PENERJEMAH
NOTASI MUSIK
MENJADI BENTUK TAMPILAN
POSISI JARI TANGAN PADA FRET GITAR



RSIF
005.1
Her
p - 1
1997

Oleh :

PAULUS HERLUKJANTO

NRP : 2691.100.015

PERPUSTAKAAN ITS	
Tgl. Terima	16-7-2003
Terima Dari	H
No. Agenda Prp.	718400

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
1997

**PERANCANGAN DAN PEMBUATAN
PERANGKAT LUNAK PENERJEMAH
NOTASI MUSIK
MENJADI BENTUK TAMPILAN
POSISI JARI TANGAN PADA FRET GITAR**

TUGAS AKHIR

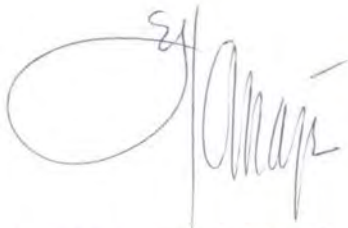
**Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik Informatika**

Pada

**Jurusan Teknik Informatika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
S u r a b a y a**

Mengetahui / Menyetujui

Dosen Pembimbing I



Ir. ESTHER HANAYA, M.Sc.

NIP. 130 816 212

Dosen Pembimbing II



Dr. Ir. HANDAYANI TJANDRASA, M.Sc.

NIP. 130 532 048

S U R A B A Y A

Agustus, 1997

ABSTRAK

Gitar adalah salah satu alat musik yang sudah memasyarakat. Hal ini disebabkan karena gitar bisa digunakan sebagai musik iringan yang praktis dan mudah dipelajari. Selain digunakan sebagai musik iringan, gitar juga bisa dimainkan solo dengan ritme dan melodi yang dibunyikan secara harmonis. Dalam menerjemahkan notasi musik ke dalam permainan gitar dibutuhkan penguasaan dan perasaan. Hal ini cukup menyulitkan bagi pemain gitar pemula. Komputer merupakan salah satu solusi untuk mengatasi masalah tersebut. Untuk itu diperlukan sebuah perangkat lunak penerjemah notasi musik ke dalam pola permainan gitar yang dapat diaplikasikan pada komputer tersebut.

Perangkat lunak ini membutuhkan masukan berupa file notasi musik yang disimpan dalam bentuk *file MIDI*, yang akan diproses untuk memperoleh posisi jari pada fret gitar untuk setiap nada yang dibunyikan. Penentuan posisi berdasarkan pola permainan yang disesuaikan dengan kemampuan perpindahan jari pada fret gitar. Proses diawali dengan membagi data nada dalam satu file menjadi beberapa interval. Berikutnya adalah penentuan posisi nada pada *fretboard* untuk setiap interval, yang dilanjutkan dengan penentuan jari untuk setiap posisi tersebut. Dari proses tersebut akan diperoleh pola perpindahan jari yang mudah direpresentasikan pada alat musik gitar.

KATA PENGANTAR

Puji syukur yang sebesar-besarnya penyusun haturkan ke hadirat Tuhan Yang Maha Esa, karena hanya dengan perkenan-Nya Tugas Akhir ini bisa terselesaikan dengan baik.

Tugas Akhir ini merupakan salah satu mata kuliah wajib dan persyaratan akademik pada jenjang pendidikan Strata-1 di Jurusan Teknik Informatika, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopembar Surabaya, untuk memperoleh gelar sarjana.

Selama persiapan dan penyusunan Tugas Akhir ini, penyusun telah banyak mendapat dukungan dan bantuan dari berbagai pihak, baik secara moril maupun materi. Untuk itu sudah sepantasnya penyusun menyampaikan ucapan terima kasih yang tidak terhingga kepada:

1. Ibu **Dr. Ir. Handayani Tjandrasa, M.Sc**, selaku Dosen Pembimbing dan Dosen Wali,
2. Ibu **Ir. Esther Hanaya, M.Sc**, selaku Dosen Pembimbing,
3. Bapak **Dr. Ir. Arief Djunaidi, M.Sc**, selaku Ketua Jurusan Teknik Informatika FTI - ITS,
4. Seluruh Dosen Jurusan Teknik Informatika FTI - ITS, atas dedikasinya selama mengajar di perkuliahan,
5. Seluruh Karyawan Jurusan Teknik Informatika FTI - ITS, atas bantuannya dalam hal kesekretariatan,
6. **Ayahanda** beserta kakak-kakak tercinta, atas doa dan dorongan semangat yang telah diberikan,
7. Yang tersayang, **Anastasia Sisy**, atas kesetiaan mendampingi,
8. Rekan **Yandis, Isaac, Awi, Dani, Kadek Patria, Kadek Wahyudi, Kadek Pasek, Igan, Tude, Bilong, Ketut Budi, Vivin, Belor, Indra** dan segenap

rekan-rekan yang namanya tidak dapat disebutkan satu persatu yang telah membantu dalam proses penyelesaian Tugas Akhir ini.

Penyusun menyadari atas segala keterbatasannya, bahwa Tugas Akhir ini masih banyak kekurangannya dan jauh dari sempurna. Untuk itu penyusun sangat mengharapkan saran dan kritik membangun dalam upaya penyempurnaan Tugas Akhir ini.

Pada akhirnya penyusun berharap semoga Tugas Akhir ini dapat memberikan manfaat bagi seluruh rekan mahasiswa / pembaca dan bagi perkembangan ilmu pengetahuan di masa-masa mendatang.

Surabaya, 26 Januari 1997

Penyusun

DAFTAR ISI

JUDUL	i
LEMBAR PENGESAHAN	ii
ABSTRAK	iii
KATA PENGANTAR	iv
DAFTAR ISI	vi
DAFTAR GAMBAR	x
BAB I PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Perumusan Masalah	2
1.3. Pembatasan Masalah	2
1.4. Tujuan	3
1.5. Metodologi	3
1.6. Sistematika Pembahasan	4
BAB II DASAR TEORI PENUNJANG	6
2.1. Musik	6
2.1.1. Umum	6
2.1.2. Notasi Musik	7
2.1.3. Tangga Nada	10
2.1.3.1. Tangga Nada Mayor	12
2.1.3.2. Tangga Nada Minor Natural	13

2.1.3.3. Tangga Nada Minor Harmonis	14
2.1.3.4. Tangga Nada Minor Melodis	15
2.1.3.5. Tangga Nada Mayor Pentatonis	15
2.1.3.6. Tangga Nada Minor Pentatonis	16
2.1.3.7. Tangga Nada Dominant Sevent Blues	16
2.1.3.8. Tangga Nada Diminished	17
2.1.4. Akord	17
2.1.4.1. Akord Mayor	18
2.1.4.2. Akord Minor	18
2.1.4.3. Akord Suspended	20
2.1.4.4. Akord Augmented	20
2.1.4.5. Akord Sixth	20
2.1.4.6. Akord Mayor Seventh	20
2.1.4.7. Akord Dominant Seventh	21
2.1.4.8. Akord Diminished	22
2.1.5. Pola Permainan Gitar	22
2.2. File MIDI	31
2.2.1. Header Chunk	31
2.2.2. Track Chunk	33
2.2.2.1. MIDI Event	35
2.2.2.2. Sysex Event	44
2.2.2.3. Meta Event	46

	2.3 Pencarian Jarak Terdekat	55
BAB III	PERENCANAAN PROGRAM	62
	3.1. Perencanaan Sistem	62
	3.1.1. Komponen Data	64
	3.1.2. Komponen Pemrosesan	65
	3.2. Perencanaan Struktur Data	67
	3.2.1. Notasi Musik dalam Format File MIDI	67
	3.2.2. Nada dalam Interval Nada Gitar	68
	3.2.3. Posisi Nada pada Fretboard	69
	3.2.4. Posisi Jari pada Fretboard	70
BAB IV	PEMBUATAN PROGRAM	71
	4.1. Umum	71
	4.2. Struktur Data	72
	4.2.1. Notasi Musik dalam Format File MIDI	72
	4.2.2. Nada dalam Interval Nada Gitar	74
	4.2.3. Posisi Nada pada Fretboard	75
	4.2.4. Posisi Jari pada Fretboard	78
	4.3. Modul Baca dan Konversi File MIDI	79
	4.4. Modul Pemilihan Posisi pada Fretboard	82
	4.5. Modul Penentuan Jari untuk Tiap Posisi	88
	4.6. Modul Pengolah Tampilan File Hasil Keluaran	95
BAB V	ANALISA PROGRAM	97

5.1. Jalan dan Keluaran Program	97
5.2. Analisa dan Evaluasi	104
BAB VI PENUTUP	107
6.1. Kesimpulan	107
6.2. Saran	108
DAFTAR PUSTAKA	110
LAMPIRAN	111

DAFTAR GAMBAR

gambar 2.1.	Bentuk dan nilai nada pada notasi balok	8
gambar 2.2.	Bentuk dan nilai tanda istirahat pada notasi balok	8
gambar 2.3.	Bentuk dan letak tanda kunci pada tuntunan nada	9
gambar 2.4.	Frekuensi nada musik delapan oktaf	11
gambar 2.5.	Tangga nada C mayor	12
gambar 2.6.	Tangga nada dengan tanda mula krus	13
gambar 2.7.	Tangga nada dengan tanda mula mol	13
gambar 2.8.	Tangga nada C minor natural	14
gambar 2.9.	Tangga nada C minor harmonis	14
gambar 2.10.	Tangga nada C minor melodis	15
gambar 2.11.	Tangga nada C mayor pentatonis	15
gambar 2.12.	Tangga nada C minor pentatonis	16
gambar 2.13.	Tangga nada C7 blues	17
gambar 2.14.	Tangga nada C diminished	17
gambar 2.15.	Gitar akustik dengan 6 senar dan 18 fret	22
gambar 2.16.	Urutan nada pada senar gitar	23
gambar 2.17.	Posisi nada pada fretboard	23
gambar 2.18.	Notasi jari tangan untuk permainan gitar	24
gambar 2.19.	Posisi nada pada fretboard untuk C Mayor Pentatonis	25
gambar 2.20.	Posisi jari pada fretboard untuk C Mayor Pentatonis	26

gambar 2.21.	Posisi nada pada fretboard untuk C Minor Pentatonis	27
gambar 2.22.	Posisi jari pada fretboard untuk C Minor Pentatonis	28
gambar 2.23.	Posisi nada pada fretboard untuk C Minor Harmonis	29
gambar 2.24.	Posisi jari pada fretboard untuk C Minor Harmonis	29
gambar 2.25.	Format File MIDI	31
gambar 2.26.	Hubungan antar posisi nada	55
gambar 2.27.	Pohon kombinasi posisi nada	57
gambar 3.1.	Diagram sistem penerjemahan notasi musik menjadi tampilan posisi jari pada fret gitar	63
gambar 4.1.	Penyalinan dan konversi data file MIDI format 1 ke dalam file penampung GT1.TMP	81
gambar 4.2.	Diagram alir proses pemilihan posisi pada fretboard	85
gambar 4.3.	Diagram alir proses pencarian jarak terpendek (Shortest Path)	86
gambar 4.4a.	Diagram alir proses penentuan posisi jari untuk akord	93
gambar 4.4b.	Diagram alir proses penentuan posisi jari untuk akord (lanjutan)	94
gambar 5.1.	Notasi musik yang akan diterjemahkan	98
gambar 5.2.	Bentuk tampilan notasi musik (nada) yang berhasil diterjemahkan	102
gambar 5.3.	Bentuk tampilan posisi jari tangan pada fretboard	103

BAB I

PENDAHULUAN

1.1. Latar Belakang

Pemanfaatan teknologi komputer dalam bidang musik dalam beberapa tahun ini menunjukkan perkembangan yang cukup pesat. Hampir semua peralatan musik elektronik dilengkapi dengan perangkat MIDI (Musical Instrument Digital Interface). Salah satu pengembangan komputer musik adalah dalam penerjemahan notasi musik.

Pada umumnya penerjemahan notasi musik ke dalam permainan gitar dilakukan secara manual dengan mengandalkan perasaan pemain. Karena untuk membunyikan satu nada dapat dilakukan lebih dari satu cara. Akibatnya untuk satu notasi musik dapat dihasilkan pola perpindahan jari yang berbeda antara pemain yang satu dengan yang lain. Keadaan tersebut kadang-kadang dapat mempengaruhi musik yang dihasilkan, terutama pada penyesuaian tempo dan dinamikanya. Hal ini bisa disebabkan oleh penempatan posisi jari pada fret gitar dan pemilihan senar yang dibunyikan kurang tepat.

Banyaknya editor musik yang sangat membantu dalam penyuntingan notasi musik, menimbulkan suatu gagasan untuk mengolah hasil keluaran dari editor musik tersebut yang secara umum disimpan dalam bentuk file MIDI.

Tugas akhir dibuat dengan harapan dapat membantu mengatasi kesulitan penerjemahan notasi musik ke dalam permainan gitar, terutama untuk pemain gitar pemula.

1.2. Perumusan Masalah

Pada alat musik gitar, satu nada memiliki relasi dengan beberapa posisi jari pada *fretboard*. Metode pencarian jarak terdekat (*Shortest Path*) merupakan satu alternatif yang dipilih untuk menentukan posisi jari pada *fretboard*. Adapun alasan pemilihan tersebut adalah dengan mempertimbangkan keterbatasan ukuran jari manusia, sehingga semakin kecil jarak posisi jari yang satu dengan yang lain pada *fretboard* akan semakin memudahkan pemain gitar.

Proses diawali dengan penentuan posisi nada pada *fretboard* dengan metode pencarian jarak terpendek. Kemudian dilanjutkan dengan penentuan jari yang digunakan, sesuai dengan posisi yang telah ditentukan.

1.3. Pembatasan Masalah

Pada tugas akhir ini data masukan yang dikenali adalah data file MIDI (*.MID) format 0 atau format 1. Data yang digunakan adalah data nada, dengan mengabaikan data-data yang berpengaruh pada efek suara.

Nada yang diproses hanyalah yang termasuk di dalam interval nada pada alat musik gitar. Jumlah nada maksimum yang dapat dibunyikan pada satu waktu sebanyak enam nada (d disesuaikan dengan banyaknya senar gitar). Apabila terdapat

nada-nada yang tidak mungkin dibunyikan secara bersamaan karena posisi nada pada *fretboard* tidak memungkinkan bagi jari tangan manusia, maka ada nada-nada yang diabaikan.

Hasil keluaran proses adalah berupa tampilan posisi jari pada fret gitar, dengan data yang disimpan dalam bentuk file dengan format khusus untuk perangkat lunak yang dibangun.

1.4. Tujuan

Tujuan tugas akhir ini adalah membangun perangkat lunak yang dapat menerjemahkan notasi musik yang telah disimpan dalam bentuk file MIDI ke dalam tampilan posisi jari pada fret gitar.

1.5. Metodologi

Metodologi yang dipakai dalam penyusunan tugas akhir ini adalah :

1. Studi literatur yang meliputi pemahaman notasi musik, pola permainan gitar, struktur file MIDI, dan metode pencarian jarak terdekat untuk penentuan posisi.
2. Perancangan antar muka visual yang menjembatani antara pemakai dan informasi yang disampaikan. Terutama karena keluaran perangkat lunak ini berupa tampilan visual.

3. Perancangan struktur data dan algoritma program. Dimana struktur data dan algoritma yang dirancang harus dapat merealisasikan perangkat lunak yang dapat menerima masukan berupa file MIDI (*.MID) format 0 atau format 1.
4. Perancangan dan pembuatan modul-modul program pembangun perangkat lunak secara terstruktur. Yaitu dimulai modul untuk membaca masukan file MIDI, kemudian modul untuk memproses data yang telah dibaca, dilanjutkan dengan modul untuk menyimpan data hasil pemrosesan data masukan, dan yang terakhir modul untuk menampilkan data keluaran.
5. Pengujian perangkat lunak guna mengetahui permasalahan yang mungkin timbul.
6. Penyempurnaan dilakukan jika masih terdapat kekurangan pada jalannya program maupun pada hasil keluarannya.
7. Pembuatan laporan.

1.6. Sistematika Pembahasan

Sistematika yang dipergunakan dalam penulisan tugas akhir ini adalah sebagai berikut:

Bab I Pendahuluan

Meliputi latar belakang masalah, perumusan masalah, pembatasan masalah, tujuan, metodologi yang digunakan, dan sistematika penyusunan naskah.

Bab II Dasar Teori Penunjang

Menguraikan teori-teori penunjang yang berhubungan dengan pembahasan notasi musik, pola permainan gitar, file MIDI dan metode penentuan posisi jari pada fret gitar.

Bab III Perencanaan Program

Menguraikan tentang perencanaan program sebagai langkah awal sebelum memulai pembuatan program. Perencanaan program yang dibahas meliputi perencanaan sistem, aliran proses dan perencanaan struktur data yang digunakan.

Bab IV Pembuatan Program

Membahas algoritma dan langkah-langkah proses yang harus dilalui sesuai dengan perencanaan program yang sudah dirancang sebelumnya.

Bab V Analisa Program

Membahas analisa program pada berbagai macam dan kondisi data masukan.

Bab VI Penutup

Memuat suatu kesimpulan dan saran atas hasil program yang dibuat berdasarkan hasil analisa program.

BAB II

DASAR TEORI PENUNJANG

Program penerjemah notasi musik ke dalam tampilan posisi jari pada fret gitar dirancang dengan tujuan untuk mempermudah dalam mengimplementasikan notasi musik ke dalam permainan gitar. Proses ini berkaitan dengan pengenalan notasi musik maupun pola permainan gitar. Untuk itu, pembahasan di dalam bab ini akan diuraikan dasar-dasar teori penunjang yang berhubungan dengan musik, file MIDI sebagai data masukan, dan metode penentuan posisi.

2.1. Musik

2.1.1. Umum

Musik adalah suatu bentuk kesenian yang merupakan hasil pengolahan suara. Dalam seni musik terdapat unsur-unsur yang harus dipenuhi, yaitu meliputi ritme, melodi, dan harmoni. Ritme artinya beraturan, yang berkaitan dengan penetapan selang waktu (tempo) dalam satu ketukan. Melodi adalah segala sesuatu yang berhubungan dengan nada, lagu, dan pola irama. Sedangkan harmoni adalah keselarasan yang merupakan perpaduan dari ritme, melodi, dinamika (kuat suara), dan timbre (warna suara).

Secara umum penulisan notasi musik ada tiga cara, yaitu berdasarkan notasi angka atau disebut juga solmisasi, notasi huruf, dan notasi balok. Selain notasi-

notasi tersebut masih banyak notasi musik lain yang kegunaannya terbatas pada alat musik tertentu.









Notasi angka hanya bisa digunakan untuk menulis lagu yang kurang dari tiga oktaf, karena jika lebih dari tiga oktaf maka penulisannya akan memiliki banyak kelemahan. Umumnya notasi ini dipakai untuk penulisan lagu yang dinyanyikan manusia atau alat musik yang memiliki frekuensi tidak lebih dari tiga oktaf. Disamping cara penulisannya mudah dan praktis, notasi angka mudah dibaca, sehingga mudah untuk dipelajari.

Notasi huruf biasanya digunakan untuk membantu penerjemahan notasi balok. Sehingga jarang sekali digunakan dalam penulisan lagu.

Notasi balok adalah notasi musik yang umum dipakai pada gubahan lagu-lagu yang lengkap. Artinya lagu yang terdiri dari perpaduan sopran, alto, tenor, bas, dan iringan. Meskipun penulisannya kurang praktis dan sukar dipelajari, notasi ini memberikan keleluasaan dalam penulisan lagu. Dengan berbagai kelebihan yang ada, notasi ini lebih banyak digunakan dalam perangkat lunak editor musik, seperti Cakewalk Professional, Music Time, dan sebagainya.









2.1.2. Notasi Musik

Ciri khas dari notasi musik umum adalah notasi balok yang mempunyai bentuk yang khusus, dan masing-masing bentuk tersebut mempunyai nilai yang tetap. Bentuk dan nilai nada maupun tanda istirahat seperti yang ditunjukkan pada gambar 2.1 dan 2.2.

Bentuk	Nilai
	1
	$\frac{1}{2}$
	$\frac{1}{4}$
	$\frac{1}{8}$
	$\frac{1}{16}$
	$\frac{1}{32}$
	$\frac{1}{64}$
	$\frac{1}{128}$

Gambar 2.1.

Bentuk dan nilai nada pada notasi balok

Bentuk	Nilai
	1
	$\frac{1}{2}$
	$\frac{1}{4}$
	$\frac{1}{8}$
	$\frac{1}{16}$
	$\frac{1}{32}$
	$\frac{1}{64}$
	$\frac{1}{128}$

Gambar 2.2.

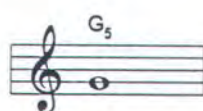
Bentuk dan nilai tanda istirahat pada notasi balok



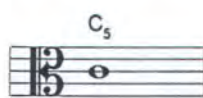
Nama dan tinggi rendahnya nada (*pitch*) pada notasi balok ditentukan oleh letaknya pada tuntunan nada yang telah dilengkapi tanda kunci tertentu^[1]. Tuntunan nada terdiri dari lima buah garis horisontal yang sejajar dan berjarak sama. Pada umumnya dipergunakan tiga buah kunci, yaitu G, C, dan F. Seperti ditunjukkan pada gambar 2.3, kunci-kunci tersebut menjelaskan :

^[1] Istilah tuntunan nada sama artinya dengan garis paranada atau sangkar nada.

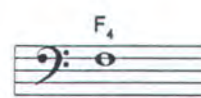
- a. Kunci G atau kunci biola menjelaskan nada G_5 yang terletak pada pusat kunci tersebut.
- b. Kunci C atau kunci gregorian menjelaskan nada C_5 yang terletak pada garis tengah.
- c. Kunci F atau kunci bas menjelaskan nada F_4 yang terletak pada garis yang diapit oleh dua buah titik.



(a) kunci G



(b) kunci C



(c) kunci F

Gambar 2.3.

Bentuk dan letak tanda kunci pada tuntunan nada^[2]

Notasi-notasi yang telah dijelaskan di atas merupakan notasi pokok. Selain notasi-notasi pokok tersebut, masih terdapat notasi-notasi yang merupakan pelengkap yang dibutuhkan dalam suatu penulisan lagu. Notasi pelengkap tersebut meliputi :

a. Tanda Birama

Yaitu tanda yang menentukan pembagian waktu sebuah lagu menjadi bagian-bagian yang sama. Pembagian tersebut dinyatakan sebagai garis birama, yaitu garis vertikal yang memotong tuntunan nada. Tanda birama ini dijelaskan dengan angka pecahan. Penyebut tanda birama menentukan nilai nada untuk satu ketukan, yang nilainya merupakan hasil pemangkatan bilangan 2. Sedangkan

^[2] Teguh Wartono dkk., *Pengantar Pendidikan Seni Musik*, Penerbit Yayasan Kanisius, 1984, pp. 24

pembilang tanda birama menentukan jumlah ketukan pada satu birama, dimana satu birama dinotasikan oleh ruang diantara dua garis birama.

b. Tanda Tempo

Adalah tanda yang menentukan lamanya satu ketukan. Tanda ini dinyatakan sebagai angka metronome, atau istilah yang mewakili tempo tertentu. Pemakaian istilah yang lazim digunakan adalah dalam bahasa Itali. Misalnya *Adagio* yang berarti tempo perlahan dengan perasaan, *Adante* yang berarti tempo biasa dan tenang, dan sebagainya.

c. Tanda Dinamik

Adalah tanda yang menjelaskan tentang keras lembutnya nada yang dibunyikan (*velocity*^[3]).

2.1.3 Tangga Nada

Nada adalah bunyi atau suara yang mempunyai frekuensi getaran tertentu. Dimana nada C adalah pokok dari nada yang lain. Pada tahun ± 1858 Akademi Seni Musik Perancis menetapkan bahwa nada A_4 disebut *nada standar* yang mempunyai frekuensi getaran 435Hz. Kemudian dari hasil kongres di Wina (1885) yaitu *International Music Congres*, hal ini dapat diakui seluruh dunia. Tetapi pada tahun 1936 menurut American Standards Association tinggi nada standar tersebut ditetapkan menjadi 440Hz^[4].

^[3] Velocity adalah istilah untuk menyatakan keras lembutnya nada pada file MIDI, yang artinya sama dengan volume nada.

^[4] Teguh Wartono dkk., *Pengantar Pendidikan Seni Musik*, Penerbit Yayasan Kanisius, 1984, pp. 41

Jarak sebuah nada dengan nada dengan frekuensi dua kali di atasnya adalah sebanyak 12 nada. Jarak tersebut disebut sebagai jarak 1 oktaf. Istilah oktaf diambil dari 8 nada yang terdapat pada tangga nada mayor. Pada gambar 2.4 ditunjukkan nada delapan oktaf mulai dari C_0 sampai C_8 beserta besar frekuensinya.

Nada	Frekuensi	Nada	Frekuensi	Nada	Frekuensi	Nada	Frekuensi
C_0	16.35	C_2	65.41	C_4	261.63	C_6	1046.50
$C_{\#0}$	17.32	$C_{\#2}$	69.30	$C_{\#4}$	277.18	$C_{\#6}$	1108.73
D_0	18.35	D_2	73.42	D_4	293.66	D_6	1174.66
$D_{\#0}$	19.45	$D_{\#2}$	77.78	$D_{\#4}$	311.13	$D_{\#6}$	1244.51
E_0	20.60	E_2	82.41	E_4	329.63	E_6	1328.51
F_0	21.83	F_2	87.31	F_4	349.23	F_6	1396.91
$F_{\#0}$	23.12	$F_{\#2}$	92.50	$F_{\#4}$	369.99	$F_{\#6}$	1479.98
G_0	24.50	G_2	98.00	G_4	392.00	G_6	1567.98
$G_{\#0}$	25.96	$G_{\#2}$	103.83	$G_{\#4}$	415.30	$G_{\#6}$	1661.22
A_0	27.50	A_2	110.00	A_4	440.00	A_6	1760.00
$A_{\#0}$	29.14	$A_{\#2}$	116.54	$A_{\#4}$	466.16	$A_{\#6}$	1864.66
B_0	30.87	B_2	123.47	B_4	493.88	B_6	1975.53
C_1	32.70	C_3	130.81	C_5	523.25	C_7	2093.00
$C_{\#1}$	34.65	$C_{\#3}$	138.59	$C_{\#5}$	554.37	$C_{\#7}$	2217.46
D_1	36.71	D_3	146.83	D_5	587.33	D_7	2349.32
$D_{\#1}$	38.89	$D_{\#3}$	155.56	$D_{\#5}$	622.25	$D_{\#7}$	2489.02
E_1	41.20	E_3	164.81	E_5	659.26	E_7	2637.02
F_1	43.65	F_3	174.61	F_5	698.46	F_7	2793.83
$F_{\#1}$	46.25	$F_{\#3}$	185.00	$F_{\#5}$	739.99	$F_{\#7}$	2959.96
G_1	49.00	G_3	196.00	G_5	783.99	G_7	3135.96
$G_{\#1}$	51.91	$G_{\#3}$	207.65	$G_{\#5}$	830.61	$G_{\#7}$	3322.44
A_1	55.00	A_3	220.00	A_5	880.00	A_7	3520.00
$A_{\#1}$	58.27	$A_{\#3}$	233.08	$A_{\#5}$	932.33	$A_{\#7}$	3729.31
B_1	61.74	B_3	246.94	B_5	987.77	B_7	3951.07
						C_8	4186.01

Gambar 2.4.

Frekuensi nada musik delapan oktaf

2.1.3.1. Tangga Nada Mayor

Tangga nada mayor adalah tangga nada normal, atau disebut juga tangga nada dasar atau natural. Tangga nada mayor mempunyai jarak antara nada dengan ketentuan $1-1-\frac{1}{2}-1-1-1-\frac{1}{2}$. Sebagai tangga nada pokok adalah tangga nada C mayor. Gambar 2.5 menunjukkan bentuk notasi tangga nada C mayor.



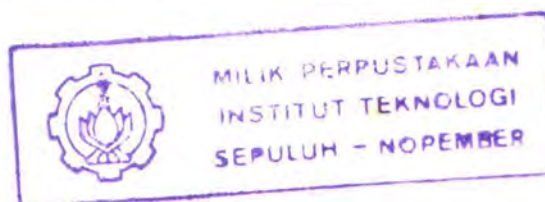
Gambar 2.5.

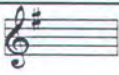


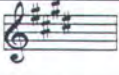



Tangga nada C mayor

Pada tangga nada pokok terdapat dua jarak $\frac{1}{2}$, yaitu antara E-F, dan antara B-C. Sehingga pada kedua jarak tersebut tidak terdapat nada tengahan. Jika E dinaikkan $\frac{1}{2}$ nada menjadi F, jika B dinaikkan $\frac{1}{2}$ nada menjadi C.

Untuk tangga nada mayor yang lain, juga harus memenuhi ketentuan aturan jarak. Maka untuk memenuhi syarat tersebut, maka beberapa nada harus dinaikkan atau diturunkan, yaitu dengan tanda-tanda *kromatis*. Tanda kromatis untuk menaikkan $\frac{1}{2}$ nada adalah \sharp (*kruis / sharp*), sedangkan untuk menurunkan $\frac{1}{2}$ nada digunakan \flat (*mol / flat*). Karena tanda ini merupakan tanda tangga nada, maka harus diletakkan di depan setelah tanda kunci dalam tuntunan tangga nada.



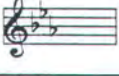
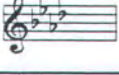
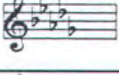
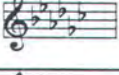
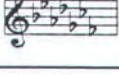
Pada gambar 2.6 ditunjukkan tangga nada perubahan yang mendapat tanda mula kruis. Sedangkan pada gambar 2.7 ditunjukkan tangga nada perubahan yang mendapat tanda mula mol. Jumlah tangga nada perubahan tersebut ada 14 tangga nada.



Notasi	Tanda Mula	Nada Dasar	Nada Perubahan
	1#	G Mayor	Fis
	2#	D Mayor	Fis, Cis
	3#	A Mayor	Fis, Cis, Gis
	4#	E Mayor	Fis, Cis, Gis, Dis
	5#	B Mayor	Fis, Cis, Gis, Dis, Ais
	6#	Fis Mayor	Fis, Cis, Gis, Dis, Ais, Eis
	7#	Cis Mayor	Fis, Cis, Gis, Dis, Ais, Eis, Bis

Gambar 2.6.

Tangga nada dengan tanda mula krus

Notasi	Tanda Mula	Nada Dasar	Nada Perubahan
	1b	F Mayor	Bes
	2b	Bes Mayor	Bes, Es
	3b	Es Mayor	Bes, Es, As
	4b	As Mayor	Bes, Es, As, Des
	5b	Des Mayor	Bes, Es, As, Des, Ges
	6b	Ges Mayor	Bes, Es, As, Des, Ges, Ces
	7b	Ces Mayor	Bes, Es, As, Des, Ges, Ces, Fes

Gambar 2.7.

Tangga nada dengan tanda mula mol

2.1.3.2. Tangga Nada Minor Natural

Tangga nada minor natural adalah tangga nada yang mempunyai ketentuan jarak $1-1\frac{1}{2}-1-1\frac{1}{2}-1-1$.

Tangga nada minor natural ini diambil dari nada ke-6 dari mayor. Maka yang disebut tangga nada minor natural pokok adalah tangga nada A minor, dimana nada A adalah nada ke-6 dari tangga nada C mayor. Pada gambar 2.8 ditunjukkan tangga nada C minor natural, yang diambil dari nada ke-6 tangga nada Es mayor dengan tanda mula $3\flat$.



Gambar 2.8.

Tangga nada C minor natural

2.1.3.3. Tangga Nada Minor Harmonis

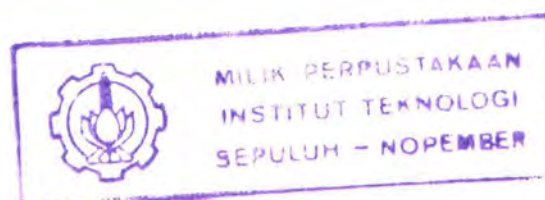
Tangga nada minor harmonis diambil dari tangga nada minor natural, dimana nada ke-7 dari tangga nada minor natural dinaikkan $\frac{1}{2}$ nada. Maka ketentuan jaraknya menjadi $1-\frac{1}{2}-1-1-\frac{1}{2}-1-\frac{1}{2}$. Pada gambar 2.9 ditunjukkan tangga nada C minor harmonis. Nada ke-7 pada tangga nada C minor natural yaitu $B\flat$, berubah menjadi B pada tangga nada C minor harmonis.



Gambar 2.9.

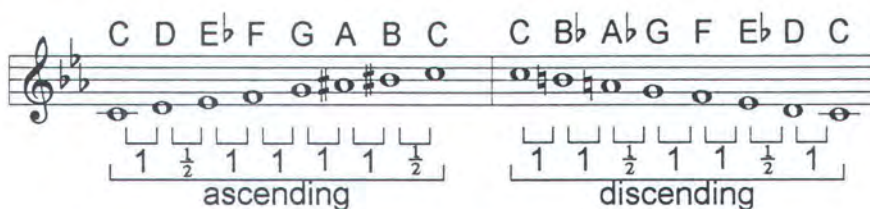
Tangga nada C minor harmonis

Tangga nada ini biasanya digunakan dalam musik Spanyol Timur Tengah dan musik klasik.



2.1.3.4. Tangga Nada Minor Melodis

Tangga nada minor melodis juga merupakan pengembangan dari tangga nada minor natural. Pada tangga nada ini, nada ke-6 dan ke-7 dari tangga nada minor natural dinaikkan $\frac{1}{2}$ nada pada urutan naik. Sedangkan pada urutan turun nada-nada tersebut dikembalikan lagi seperti semula. Maka aturan jarak untuk tangga nada ini adalah $1-\frac{1}{2}-1-1-1-\frac{1}{2}$ untuk urutan naik (*ascending*), dan $1-1-\frac{1}{2}-1-1-\frac{1}{2}-1$ untuk urutan turun (*descending*). Pada gambar 2.10 ditunjukkan tangga nada C minor melodis.

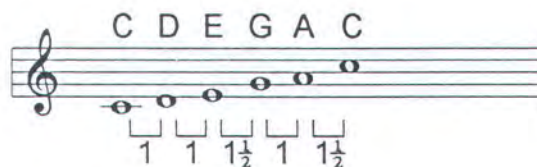


Gambar 2.10.

Tangga nada C minor melodis

2.1.3.5. Tangga Nada Mayor Pentatonis

Tangga nada mayor pentatonis adalah tangga nada yang memiliki 5 macam nada, dimana nada tersebut diambil dari nada ke-1, ke-2, ke-3, ke-5, dan ke-6 dari tangga nada mayor. Maka ketentuan jarak dari tangga nada ini adalah $1-1-1\frac{1}{2}-1-1\frac{1}{2}$. Pada gambar 2.11 ditunjukkan tangga nada C mayor pentatonis.

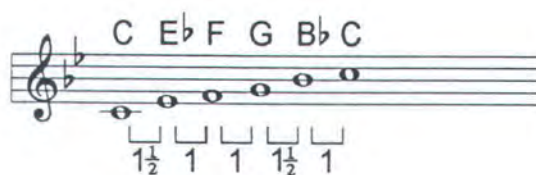


Gambar 2.11.

Tangga nada C mayor pentatonis

2.1.3.6. Tangga Nada Minor Pentatonis

Tangga nada mayor pentatonis adalah tangga nada yang memiliki 5 macam nada, dimana nada tersebut diambil dari nada ke-1, ke-3, ke-4, ke-5, dan ke-7 dari tangga nada minor natural, atau diambil dari nada ke-5 pada tangga nada mayor pentatonis. Karena sebagaimana diketahui bahwa nada ke-5 tangga nada mayor pentatonis diambil dari nada ke-6 tangga nada mayor, dan tangga nada minor natural diambil dari nada ke-6 tangga nada mayor. Maka ketentuan jarak dari tangga nada ini adalah $1\frac{1}{2}$ -1-1- $1\frac{1}{2}$ -1. Pada gambar 2.12 ditunjukkan tangga nada C minor pentatonis.



Gambar 2.12.

Tangga nada C minor pentatonis

2.1.3.7. Tangga Nada Dominant Seventh Blues

Tangga nada dominant seventh blues adalah tangga nada yang digunakan sebagai unsur penyusun musik blues, yaitu satu aliran dari musik jazz. Tangga nada ini diambil dari tangga nada mayor, dimana nada ke-7 diturunkan $\frac{1}{2}$ nada. Sehingga ketentuan jarak dari tangga nada ini adalah 1-1- $\frac{1}{2}$ -1-1- $\frac{1}{2}$ -1. Tangga nada ini digunakan sebagai pedoman pembentukan akord dominant seventh. Pada gambar 2.13 ditunjukkan tangga nada C7 blues.

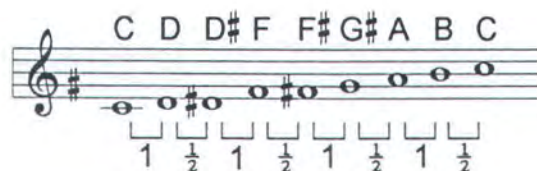


Gambar 2.13.

Tangga nada C7 blues

2.1.3.8. Tangga Nada Diminished

Tangga nada diminished adalah tangga nada yang memiliki ketentuan jarak $1-\frac{1}{2}-1-\frac{1}{2}-1-\frac{1}{2}-1-\frac{1}{2}$. Karena bentuk jaraknya yang unik, maka perubahan nada dasar ke nada ke-3, ke-5, dan ke-7 tidak terdapat perubahan tanda kromatis. Sehingga hanya memiliki 3 bentuk, yaitu A-B-C-D-D \sharp -F-F \sharp -G \sharp -A (bentuk pertama dengan nada dasar A), B-C \sharp -D-E-F-G-G \sharp -A \sharp -B (bentuk kedua dengan nada dasar B), dan C \sharp -D \sharp -E-F \sharp -G-A-A \sharp -B \sharp -C \sharp (bentuk ketiga dengan nada dasar C \sharp). Pada gambar 2.14 ditunjukkan tangga nada diminished bentuk pertama dengan nada dasar C.



Gambar 2.14.

Tangga nada C diminished

2.1.4. Akord

Akord adalah paduan nada-nada yang selaras dan dibunyikan pada saat yang sama. Secara umum akord digunakan sebagai iringan sebuah melodi. Pada dasarnya akord dibentuk dari tangga nada tertentu.

Sebagaimana tangga nada, akord pokok adalah akord mayor. Dari akord dasar ini dapat dikembangkan menjadi akord-akord yang lain. Berikut ini adalah penjelasan tentang macam dan susunan nada pembentuk akord.

2.1.4.1. Akord Mayor

Akord mayor atau akord pokok dibentuk oleh nada pertama, ketiga, dan kelima dalam tangga nada mayor. Misalnya pada tangga nada C mayor berikut ini :

1	2	3	4	5	6	7	1
C	D	E	F	G	A	B	C

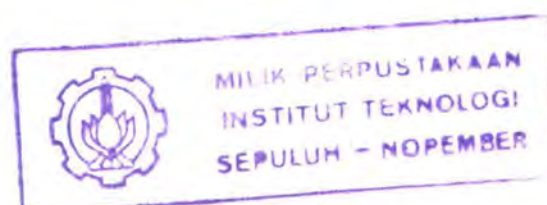
Berdasarkan tangga nada tersebut maka akord C mayor adalah 1 3 5
C E G

Demikian juga pada akord mayor yang lain, dapat dicari dengan cara yang sama.

Untuk iringan sebuah melodi dibutuhkan pasangan akord. Pada tangga nada mayor terdapat pasangan akord mayor, yaitu akord pertama (akord dasar), keempat, dan kelima. Sehingga pasangan akord mayor pada tangga nada C mayor adalah akord C mayor, F mayor, dan G mayor. Tanpa mengubah tangga nada, dengan menganggap F sebagai nada pertama maka akord F mayor adalah F-A-C. Demikian juga dengan cara yang sama, dapat diketahui akord G mayor adalah G-B-D. Susunan nada F mayor maupun G mayor tersebut sesuai dengan susunan yang dibentuk berdasarkan tangga nadanya masing-masing.

2.1.4.2. Akord Minor

Sebagaimana akord mayor, akord minor juga dibentuk oleh nada pertama, ketiga, dan kelima dalam tangga nada minor. Misalnya pada tangga nada A minor :



1	2	3	4	5	6	7	1
A	B	C	D	E	F	G	A

Berdasarkan tangga nada tersebut maka akord A minor adalah 1 3 5
A C E

Pada tangga nada minor juga terdapat pasangan akord minor. Sesuai dari bentuk asli tangga nada minor, dimana nada pertamanya diambil dari nada keenam pada tangga nada mayor, maka untuk tangga nada C mayor akord dasar minornya adalah akord A minor (A adalah nada keenam pada tangga nada C mayor). Pasangan akord minor juga diambil dari akord pertama (akord dasar), keempat, dan kelima. Sehingga pasangan akord minor pada tangga nada A minor adalah akord A minor, D minor, dan E minor. Tanpa mengubah tangga nada, dengan menganggap D adalah nada pertama maka dapat diketahui bahwa akord D minor adalah D-F-A. Sedangkan akord kelima adalah akord E minor dengan susunan E-G-B.

Pada tangga nada minor harmonis pasangan akordnya hampir sama dengan yang ada pada tangga nada minor natural. Perbedaannya pada tangga nada minor harmonis akord kelima adalah akord mayor. Hal ini disebabkan nada ketujuh pada tangga nada minor harmonis lebih tinggi setengah nada daripada nada ketujuh pada tangga nada minor natural. Sehingga akord kelima pada tangga nada A minor harmonis adalah akord E mayor dengan susunan E-G[#]-B.

Untuk mencari akord minor dari akord mayor yang sudah ada, bisa dilakukan dengan cara yang lebih mudah, yaitu dengan menurunkan nada ketiga

setengah nada. Misalnya akord C mayor adalah C-E-G, maka akord C minor (Cm) adalah C-E \flat -G.

2.1.4.3. Akord Suspended

Akord suspended adalah akord pokok mayor yang nada ketiganya dinaikkan setengah nada. Hal ini sama dengan mengganti nada ketiga dengan nada keempat pada tangga nada mayor. Misalnya akord C mayor adalah C-E-G, maka akord C suspended (Csus atau Csus4) adalah C-E \sharp -G atau C-F-G.

2.1.4.4. Akord Augmented

Akord augmented adalah akord pokok mayor yang nada kelimanya dinaikkan setengah nada. Misalnya akord C mayor adalah C-E-G, maka akord C augmented (Caug atau C+) adalah C-E-G \sharp .

2.1.4.5. Akord Sixth

Akord sixth adalah akord pokok ditambah dengan nada keenam pada tangga nada mayor. Sehingga untuk akord C6 adalah C-E-G-A.

2.1.4.6. Akord Mayor Seventh

Akord mayor seventh adalah akord pokok ditambah dengan nada ketujuh pada tangga nada mayor. Sehingga untuk akord C mayor seventh (CM7) adalah C-E-G-B.

Akord mayor seventh juga digunakan sebagai dasar untuk pengembangan akord yang lain seperti akord mayor 9th, akord mayor 11th, dan akord mayor 13th. Misalnya pada tangga nada C mayor berikut :

1	2	3	4	5	6	7	8	9	10	11	12	13
C	D	E	F	G	A	B	C	D	E	F	G	A

Akord C mayor 9th (CM9) adalah 1 3 5 7 9
 C E G B D

Akord C mayor 11th (CM11) adalah 1 3 5 7 11
 C E G B F

Akord C mayor 13th (CM13) adalah 1 3 5 7 13
 C E G B A

Demikian juga pada akord mayor yang lain, dapat dicari dengan cara yang sama.

2.1.4.7. Akord Dominant Seventh

Akord dominant seventh adalah akord pokok ditambah dengan nada ketujuh pada tangga nada mayor yang telah diturunkan setengah nada. Sehingga untuk nada C dominant seventh (C7) adalah C-E-G-B \flat .

Sebagaimana akord mayor seventh, akord dominant seventh digunakan sebagai dasar pengembangan untuk akord 9th, akord 11th, dan akord 13th. Yaitu dengan cara menurunkan nada ketujuh setengah nada. Sehingga untuk akord C9 adalah C-E-G-B \flat -D, akord C11 adalah C-E-G-B \flat -F, dan akord C13 adalah C-E-G-B \flat -A.

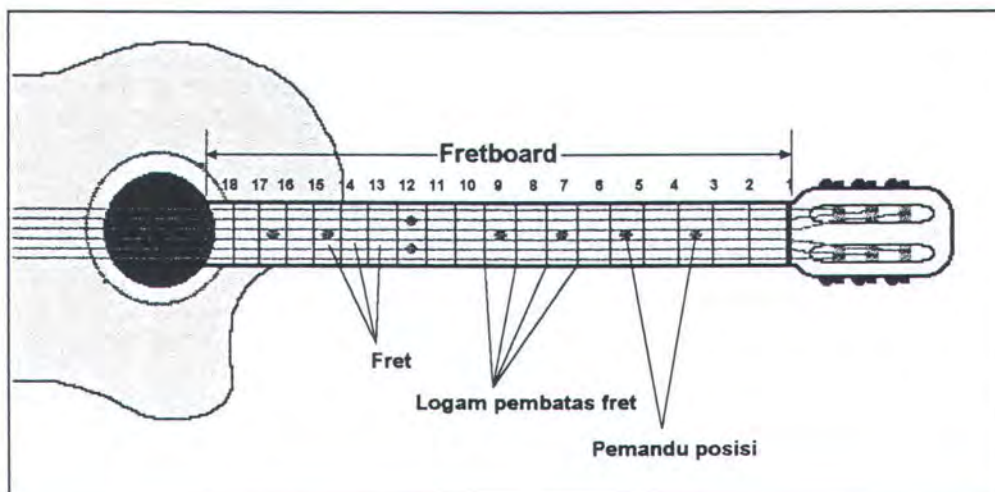
Selain akord-akord pengembangan tersebut, masih banyak akord-akord pengembangan yang lain. Adapun perubahan dan penamaan akord berdasarkan pada nada penyusunnya. Seperti akord C6+9 berarti akord C6 ditambah nada kesembilan pada tangga nada C mayor. Contoh yang lain misalnya akord C7-9 berarti akord C7 ditambah nada kesembilan pada tangga nada C mayor yang telah diturunkan setengah nada.

2.1.4.8. Akord Diminished

Akord diminished adalah akord yang dibentuk oleh nada pertama, ketiga, kelima, dan ketujuh pada tangga nada diminished. Seperti telah dijelaskan bahwa tangga nada diminished hanya ada 3 bentuk, yaitu A-B-C-D-D \sharp -F-F \sharp -G \sharp -A, B-C \sharp -D-E-F-G-G \sharp -A \sharp -B, dan C \sharp -D \sharp -E-F \sharp -G-A-A \sharp -B \sharp -C \sharp . Dimana perubahan nada dasar tidak menyebabkan perubahan tanda kromatis terhadap salah satu dari keduanya. Sehingga akord yang terbentuk juga terbatas pada 3 macam, yaitu A-C \sharp -D \sharp -F \sharp (akord Adim / Cdim / D \sharp dim / F \sharp dim), B-D-F-G \sharp (akord Bdim / Ddim / Fdim / G \sharp dim), dan C \sharp -E-G-A \sharp (akord C \sharp dim / Edim / Gdim / A \sharp dim).

2.1.5. Pola Permainan Gitar

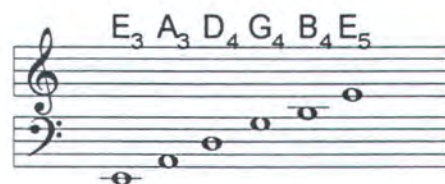
Gitar adalah alat musik dengan senar yang dipetik dan dilengkapi fret. Fret adalah kisi-kisi di antara potongan-potongan logam tipis pada tangkai gitar (lihat gambar 2.15), dimana untuk selanjutnya dalam pembahasan ini disebut sebagai *fretboard*.



Gambar 2.15.

Gitar akustik dengan 6 senar dan 18 fret

Pada umumnya jumlah senar pada gitar ada 6 buah, sedangkan jumlah fret maksimal 24 buah, dan dapat menghasilkan 49 nada yaitu E_3 sampai dengan E_7 , atau sebanyak 4 oktaf nada. Pada gambar 2.16 ditunjukkan urutan nada senar gitar. Penomoran senar gitar dimulai dari senar yang paling bawah atau paling kecil, dimana dihasilkan nada yang paling tinggi. Maka nada E_3 untuk senar no.6, A_3 untuk no.5, D_4 untuk no.4, G_4 untuk no.3, B_4 untuk no.2, dan E_5 untuk no.1.



Gambar 2.16.

Urutan nada pada senar gitar

Perubahan nada untuk perpindahan satu fret adalah $\frac{1}{2}$ nada. Dengan berpedoman pada nada senar pada fret ke-0 (tanpa penekanan), maka dapat digambarkan nada-nada pada *fretboard* seperti yang ditunjukkan pada gambar 2.17.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
E	D#	D	C#	C	B	A#	A	G#	G	F#	F	E	D#	D	C#	C	B	A#	A	G#	G	F#	F	E_3
5	5	5	5	5	4	4	4	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	A_3
A	G#	G	F#	F	E	D#	D	C#	C	B	A#	A	G#	G	F#	F	E	D#	D	C#	C	B	A#	D_4
5	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	4	4	4	4	4	4	3	3	D_4
D	C#	C	B	A#	A	G#	G	F#	F	E	D#	D	C#	C	B	A#	A	G#	G	F#	F	E	D#	G_4
6	6	6	5	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	4	4	4	G_4
G	F#	F	E	D#	D	C#	C	B	A#	A	G#	G	F#	F	E	D#	D	C#	C	B	A#	A	G#	B_4
6	6	6	6	6	6	6	6	5	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	B_4
B	A#	A	G#	G	F#	F	E	D#	D	C#	C	B	A#	A	G#	G	F#	F	E	D#	D	C#	C	E_5
6	6	6	6	6	6	6	6	6	6	6	6	5	5	5	5	5	5	5	5	5	5	5	5	E_5
E	D#	D	C#	C	B	A#	A	G#	G	F#	F	E	D#	D	C#	C	B	A#	A	G#	G	F#	F	
7	7	7	7	7	6	6	6	6	6	6	6	6	6	6	6	6	5	5	5	5	5	5	5	

Gambar 2.17.

Posisi nada pada *fretboard*

Posisi nada pada *fretboard* tersebut sudah merupakan ketetapan. Apabila pada kenyataannya terdapat gitar yang memiliki nada yang kurang tepat pada fret tertentu, biasanya disebabkan oleh ukuran fret yang tidak standar.

Disamping notasi posisi nada, pada alat musik gitar juga terdapat ketentuan untuk posisi atau cara menekan maupun memetik senar gitar. Notasi tersebut berkaitan dengan jari yang digunakan. Gambar 2.18 menjelaskan notasi jari untuk permainan gitar. Secara umum jari yang digunakan untuk menekan senar pada *fretboard* adalah jari tangan kiri, sedangkan untuk memetik senar digunakan jari tangan kanan. Pada tugas akhir ini hanya memperhatikan notasi tangan kiri, yaitu yang digunakan pada *fretboard*^[5].



Gambar 2.18.

Notasi jari tangan untuk permainan gitar^[6]

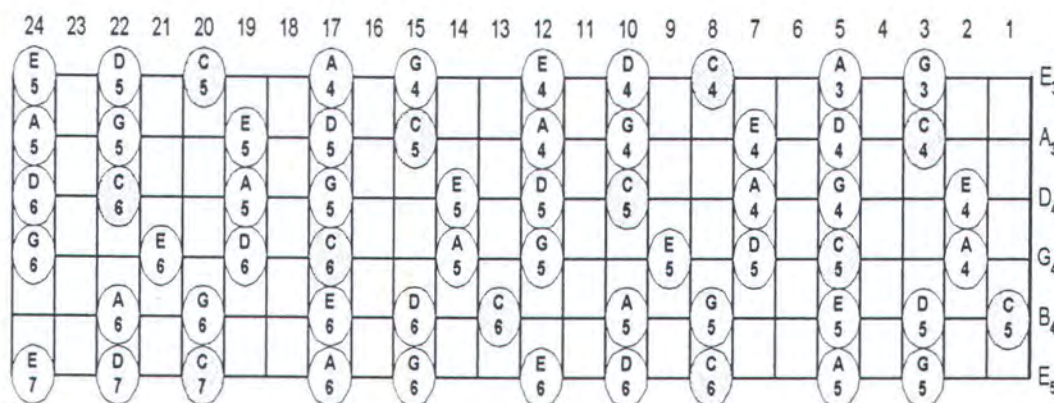
^[5] Untuk pemain "left handed" (kidal) notasi jari yang digunakan adalah kebalikan dari notasi umum (right handed). Sehingga notasi tangan kiri untuk right handed sama dengan notasi tangan kanan untuk left handed.

^[6] Francis A. Rozells & Clifford Cheam, **Rock & Heavy Metal Guitar**, Penerbit Muzikal, 1994, pp. 7

Pola melodi permainan gitar tidak memiliki pedoman khusus dalam hal penentuan posisi nada maupun jari yang digunakan. Yang paling utama adalah bahwa posisi yang digunakan mudah diimplementasikan, dengan ukuran jari yang terbatas^[7]. Berikut ini adalah beberapa contoh pola melodi pada permainan gitar. Pola melodi tersebut meliputi pola melodi pada tangga nada C mayor pentatonis, C minor pentatonis, dan C minor harmonis.

a. Pola Melodi pada Tangga Nada C Mayor Pentatonis

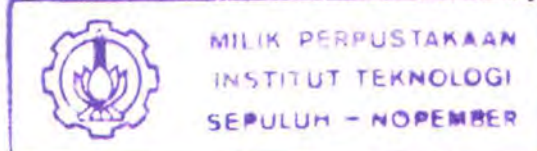
Nada-nada pada tangga nada C mayor pentatonis adalah C, D, E, G, dan A. Posisi nada-nada tersebut pada *fretboard* seperti yang ditunjukkan pada gambar 2.19.



Gambar 2.19.

Posisi nada pada *fretboard* untuk C Mayor Pentatonis

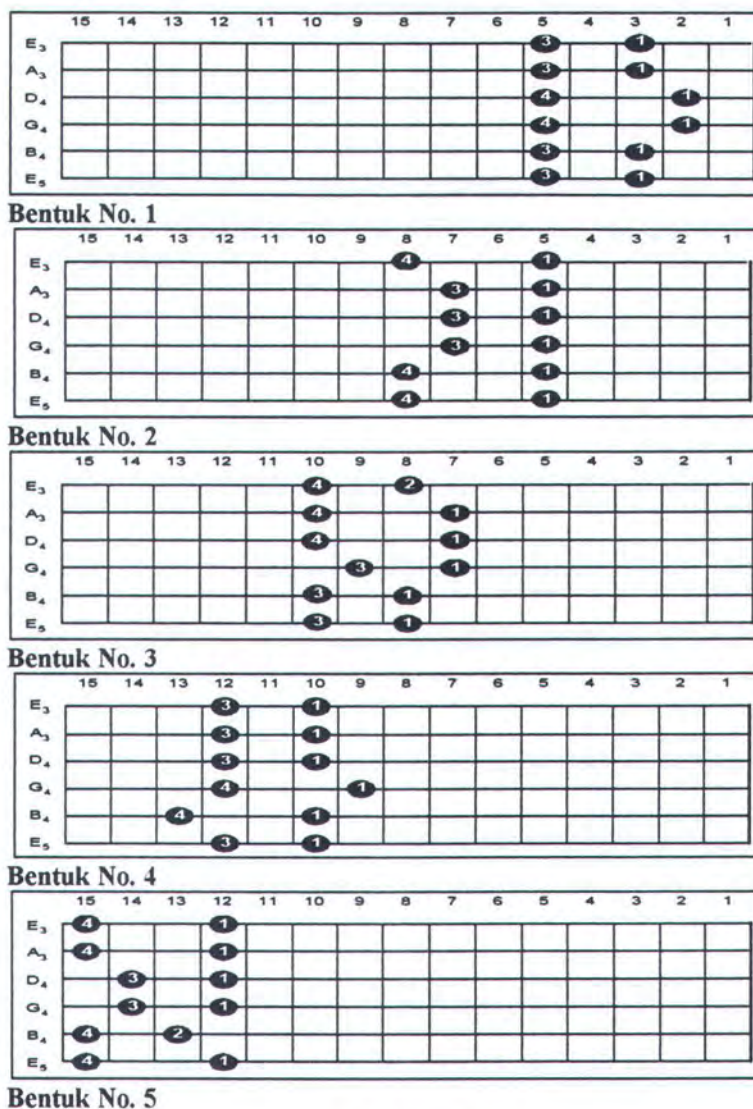
Penentuan jari yang digunakan pada posisi-posisi tersebut, tergantung pada batas nada terendah dan tertinggi yang digunakan dalam suatu melodi. Adakalanya



^[7] Jika pada saat bersamaan jari no.1 pada fret ke-1 dan jari no.4 pada fret ke-5 untuk gitar dengan ukuran fret standar, maka ukuran jari dianggap sebagai ukuran standar. Jika tidak dapat memenuhi kriteria tersebut maka bisa menggunakan gitar khusus dengan ukuran fret yang disesuaikan dengan ukuran jari pemakai.

pola perpindahan jari *ascending* (urutan naik) berbeda dengan *descending* (urutan turun), untuk nada-nada yang sama.

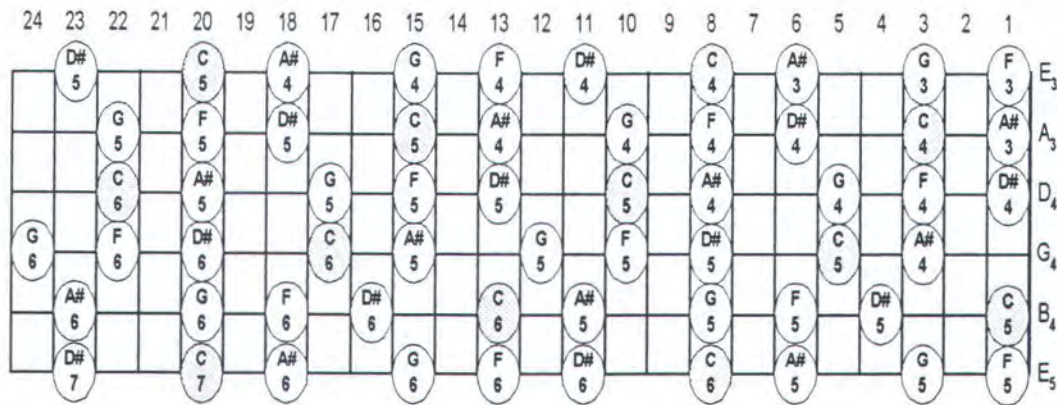
Pada gambar 2.20 ditunjukkan pola penempatan posisi jari untuk C mayor pentatonis.



Gambar 2.20.

Posisi jari pada *fretboard* untuk C Mayor Pentatonis^[8]

^[8] Francis A. Rozells & Clifford Cheam, *Rock & Heavy Metal Guitar*, Penerbit Muzikal, 1994, pp.26



Gambar 2.21.

Posisi nada pada *fretboard* untuk C Minor Pentatonis

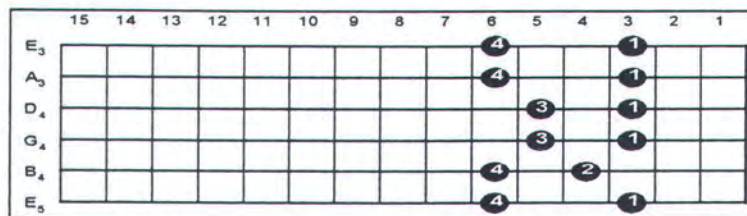
b. Pola Melodi pada Tangga Nada C Minor Pentatonis

Pola melodi pada tangga nada C minor pentatonis adalah pola melodi yang terdiri dari nada-nada C, E_b , F, G, dan B_b . Seperti halnya pola melodi pada tangga nada C mayor pentatonis, pola melodi ini memiliki posisi tertentu pada *fretboard*. Posisi nada-nada tersebut pada *fretboard* seperti yang ditunjukkan pada gambar 2.21, sedangkan pola penempatan posisi jari ditunjukkan pada gambar 2.22.

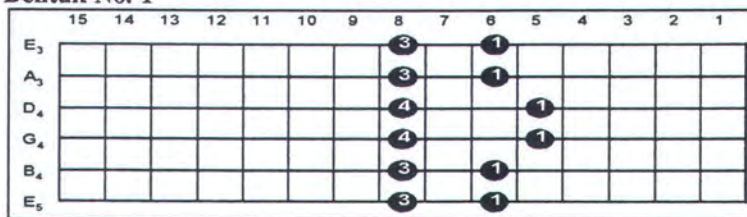
c. Pola Melodi pada Tangga nada C Minor Harmonis

Contoh pola melodi yang ketiga adalah pola melodi pada tangga nada C Minor Harmonis. Nada-nada yang terdapat pada tangga nada tersebut adalah C, D, E_b , F, G, A_b , dan B. Posisi nada-nada tersebut pada *fretboard* seperti yang ditunjukkan pada gambar 2.23.

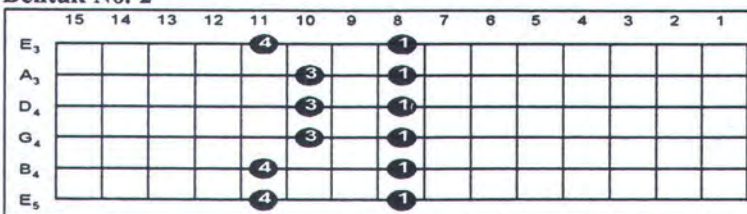
Pola penempatan posisi jari pada tangga nada C minor harmonis lebih lebar dari pada kedua contoh tangga nada pentatonis yang telah dijelaskan. Jika pada tangga nada pentatonis lebar posisi penempatan jari rata-rata adalah 4 fret, sedangkan pada tangga nada minor harmonis lebar posisi penempatan jari rata-rata adalah 5 fret.



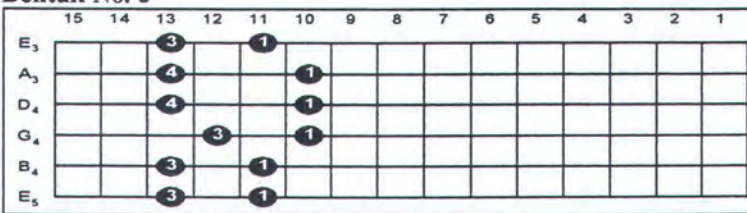
Bentuk No. 1



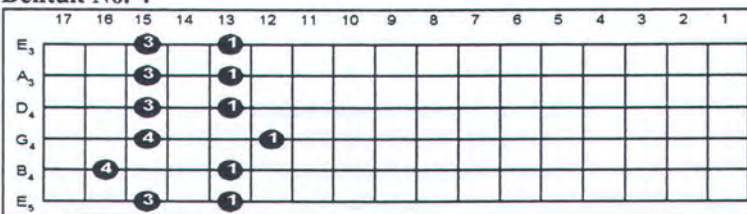
Bentuk No. 2



Bentuk No. 3



Bentuk No. 4



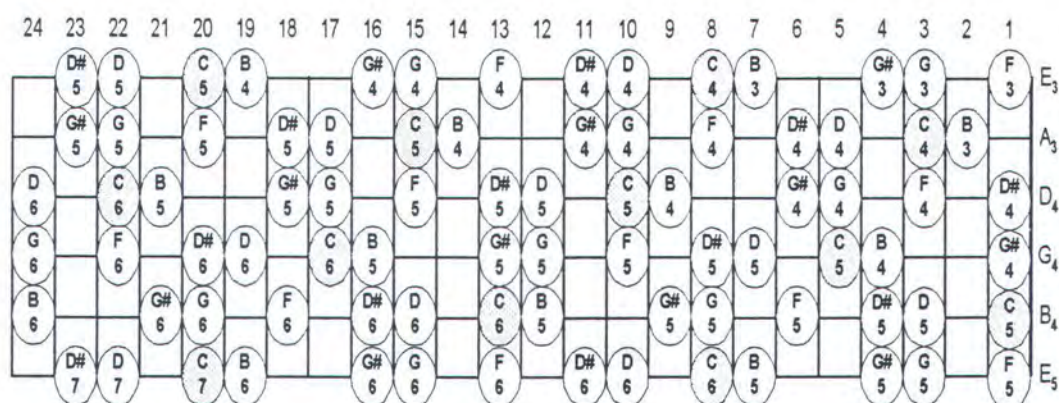
Bentuk No. 5

Gambar 2.22.

Posisi jari pada *fretboard* untuk C Minor Pentatonic^[9]

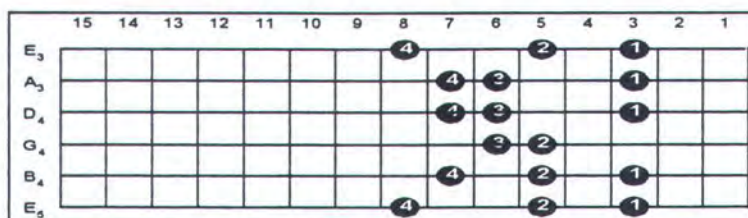
Beberapa pola penempatan posisi jari pada tangga nada C minor harmonis seperti yang ditunjukkan pada gambar 2.24. Dari gambar tersebut terdapat empat bentuk pola perpindahan jari, yaitu no.1 untuk nada G₃ sampai dengan B₅, no.2 untuk nada B₃ sampai dengan D₆, no.3 untuk nada D₄ sampai dengan F₆, dan yang terakhir no.4 untuk nada F₄ sampai dengan G₆[#].

^[9] Francis A. Rozzels & Clifford Cheam, **Rock & Heavy Metal Guitar**, Penerbit Muzikal, 1994, pp. 27

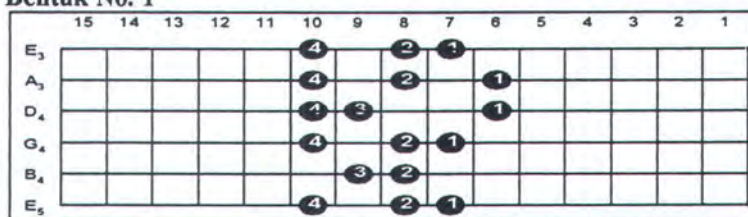


Gambar 2.23.

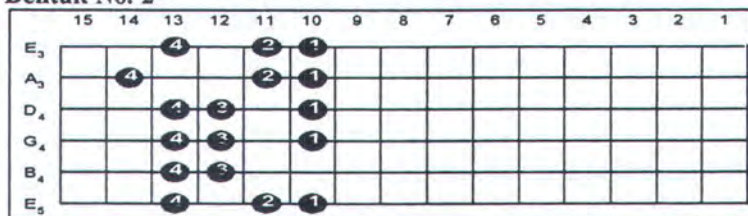
Posisi nada pada *fretboard* untuk C Minor Harmonis



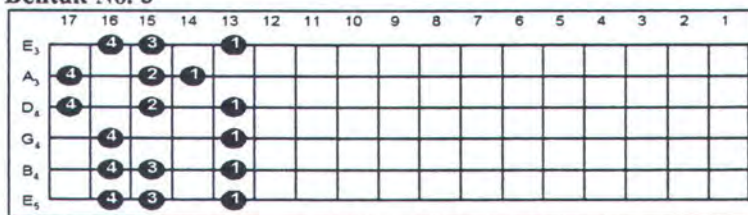
Bentuk No. 1



Bentuk No. 2



Bentuk No. 3



Bentuk No. 4

Gambar 2.24.

Posisi jari pada *fretboard* untuk C Minor Harmonis^[10]

^[10] Francis A. Rozells & Clifford Cheam, *Rock & Heavy Metal Guitar*, Penerbit Muzikal, 1994, pp. 28

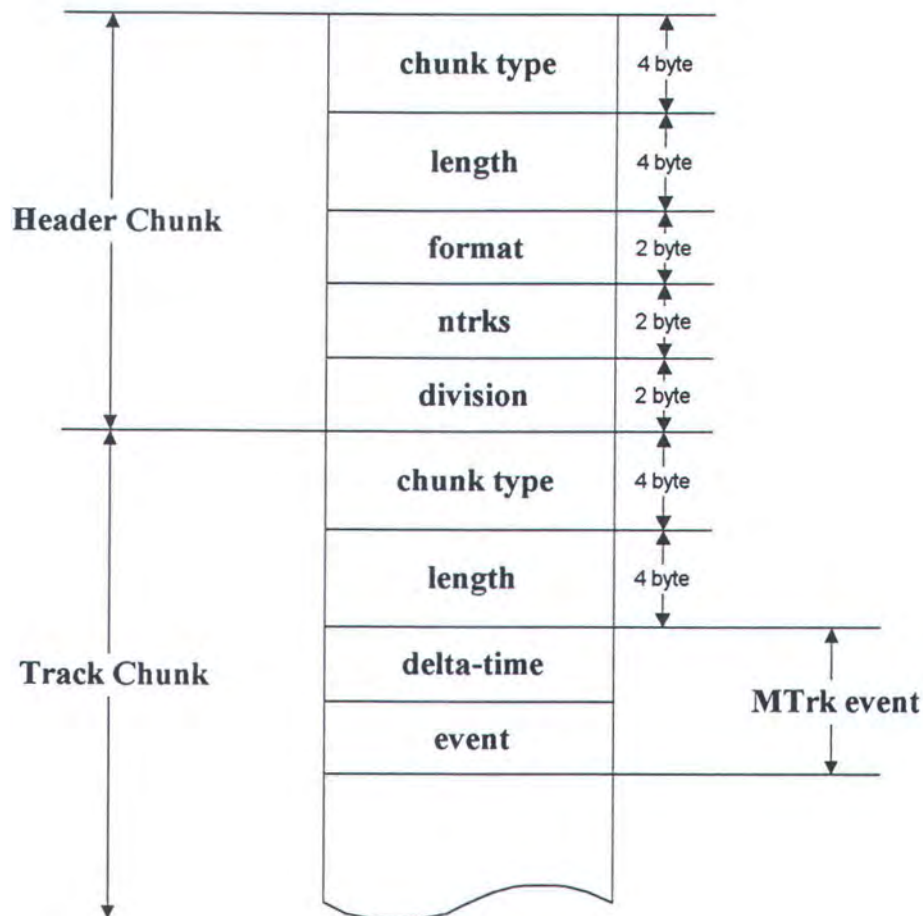
Pola melodi yang telah dijelaskan di atas juga digunakan sebagai dasar penempatan jari pada *fretboard* untuk akord. Sebagaimana telah dijelaskan bahwa akord merupakan unsur harmonis yang digunakan sebagai iringan melodi. Penggabungan unsur harmonis (akord) dan unsur melodis sering direpresentasikan dalam suatu permainan gitar tunggal. Dalam hal ini akord dinotasikan sebagai unsur melodis, dimana biasanya terdapat nada-nada tertentu dalam akord tersebut yang diabaikan karena keterbatasan posisi-posisi nada yang bisa diimplementasikan dalam permainan gitar.

Selain unsur-unsur tersebut, pada permainan gitar masih terdapat unsur-unsur lain yang menyangkut teknik memetik senar maupun teknik penempatan jari pada *fretboard* untuk menghasilkan efek suara tertentu, seperti *crosspicking*, *sweep picking*, *hammer-ons*, *pull-offs*, *string bending*, *harmonics*, *tapping* (teknik dua tangan), dan berbagai istilah lainnya. Adapun unsur-unsur tersebut tidak diuraikan pada pembahasan tugas akhir ini, karena menyangkut unsur yang berpengaruh pada efek suara yang tidak termasuk dalam batasan permasalahan tugas akhir ini.

Pembahasan pola permainan gitar ini lebih ditekankan pada pola penempatan jari pada *fretboard* terhadap unsur melodis, karena unsur tersebut yang digunakan sebagai dasar metode penentuan posisi jari pada *fretboard*.

2.2. File MIDI

Dalam tugas akhir ini file notasi musik yang bisa diterima sebagai masukan adalah file MIDI (*.MID). File MIDI ini mempunyai struktur seperti pada gambar 2.25 berikut :



Gambar 2.25.
Format File MIDI^[11]

2.2.1. Header Chunk

Header chunk pada awal file berisikan informasi spesifikasi dasar tentang data yang ada dalam file. Susunan header chunk secara lengkap adalah sebagai berikut :

^[11] Neil Rowland, Jr., "MIDI" Help File, 1993

<Header Chunk> = <chunk type><lenght><format><ntrks><division>

Sesuai struktur di atas **<chunk type>** adalah empat karakter ASCII 'Mthd', **<lenght>** adalah data 32-bit yang berisi angka konstanta 6 (panjang data berikutnya 6 byte).

Bagian data terdiri dari tiga kali 16-bit (word). 16-bit pertama berisi **<format>**, yang menjelaskan organisasi data dari file. Hanya ada tiga nilai **<format>**, yaitu :

- 0 - file terdiri atas sebuah *multi-channel track*
- 1 - file terdiri atas satu atau lebih *simultaneous track* yang berurutan
- 2 - file terdiri atas satu atau lebih bentuk *sequentialy independent single-track*

File format 0 memiliki sebuah header chunk yang diikuti oleh sebuah track chunk, merupakan bentuk paling sederhana dan mudah direpresentasikan. Sedangkan format 1 dan 2 memiliki sebuah header chunk yang diikuti oleh satu atau lebih track chunk. Format 1 berbentuk *verticaly one-dimentional* (satu dimensi vertikal), dengan data yang disimpan dalam track-track yang simultan (dibaca secara bersamaan). Format 2 berbentuk *horizontaly one-dimentional*, dengan data yang disimpan dalam track-track yang independen dan sequensial (dibaca secara berurutan).

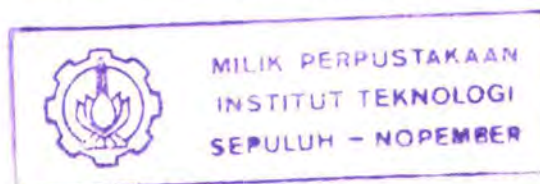
Bagian 16-bit selanjutnya **<ntrks>**, adalah jumlah track chunk dalam file. Untuk format 0 nilainya selalu satu.

Bagian 16-bit yang ketiga <**division**>, menjelaskan arti dari *delta-time*. Terdapat dua format, yang pertama untuk waktu metrik (standar), dan yang kedua untuk waktu *time-code-based*. Penjelasannya adalah sebagai berikut :

0	ketukan tiap nada seperempat			
1	format negatif SMPTE		ketukan tiap frame	
15	14	8	7	0

Jika bit 15 dari <**division**> adalah 0, maka bit 14 sampai 0 menunjukkan jumlah ketukan *delta time* yang dihitung dari nilai nada seperempat. Misalnya, jika <**division**> adalah 96, maka interval waktu dari nada seperdelapan di antara dua *event* dalam file adalah 48.

Jika bit 15 dari <**division**> adalah 1, *delta time* dalam file berhubungan dengan pembagian dari satu detik, yang mana sesuai dengan *SMPTE (Society for Motion Picture and Television Engineer)* dan *MIDI Time Code*. Bit 14 sampai 8 berisi salah satu dari empat nilai -24, -25, -29, atau -30, berhubungan dengan empat standar format SMPTE dan MIDI Time Code (-29 berhubungan dengan 30 drop frame), dan menunjukkan jumlah frame tiap detik. Bentuk angka negatif tersebut dinyatakan dalam bentuk *two's compliment*. Byte kedua (angka positif) adalah resolusi tanpa frame : nilainya mungkin 4 (MIDI Time Code resolusi), 8, 10, 80 (bit resolusi), atau 100.



2.2.2. Track Chunk

Track chunk adalah tempat dimana data-data notasi musik disimpan. Setiap track chunk mengandung rangkaian *event*, diselingi dengan *delta-time*. Format dari

track chunk sama untuk semua format file MIDI (format 0, 1, atau 2). Susunan track chunk adalah sebagai berikut (+ berarti satu atau lebih) :

<Track Chunk> = <chunk type><length><MTrk event>+

Susunan dari MTrk event sebagai berikut :

<Mtrk event> = <delta-time><event>

<delta time> adalah sebagai *variable-length quantity*. Data ini menjelaskan jumlah waktu (delay) sebelum event berikutnya. Adapun yang dimaksud *variable-length quantity* adalah sebagai berikut :

Suatu nilai yang direpresentasikan 7 bit tiap byte, *most significant bits first*.

Bit 7 pada semua byte diset 1, kecuali pada byte terakhir atau terkecil, bit 7 diset 0. Dalam hal ini jika nilainya adalah antara 0-127 maka dinyatakan dalam satu byte. Jika lebih dari 127, maka dinyatakan sebagai bilangan basis 128 ditambah dengan nilai ketentuan setting. Misalnya bilangan tersebut adalah 8F (143), maka nilainya pada basis 128 adalah 01 0F. Setelah itu nilai tersebut ditambah dengan nilai ketentuan setting yaitu 80 00, sehingga nilainya dalam *variable-length quantity* adalah 81 0F. Beberapa contoh berikut ini merupakan representasi dari *variable-length quantity* :

nilai sebenarnya	variable-length quantity
0000007F	7F
00000080	81 00
00003FFF	FF 7F
00004000	81 80 00
001FFFFFFF	FF FF 7F
00200000	81 80 80 00
0FFFFFFF	FF FF FF 7F

Pada event awal suatu track yang diproses paling awal, atau jika dua event diproses secara bersamaan, maka digunakan *delta-time* 0 (*delta-time* selalu digunakan meskipun nilainya 0). Delta-time dinyatakan sebagai bagian dari satu ketukan (atau satu detik untuk waktu dengan format SMPTE), sesuai spesifikasi pada header chunk.

Sedangkan **<event>** mempunyai beberapa kemungkinan sebagai berikut :

<event> = <MIDI event> | <sysex event> | <meta-event>

2.2.2.1. MIDI Event

<MIDI event> berisi data informasi *MIDI channel message*. *Running status* digunakan jika byte *status* pada MIDI channel message sama dengan status event sebelumnya (untuk MIDI event yang berurutan). Dimana status event harus dispesifikasikan untuk setiap event awal pada track chunk. Delta-time tidak hanya berlaku untuk sebuah event saja, tetapi juga berlaku untuk event-event sebelumnya, tergantung dari status masing-masing event tersebut. Dengan demikian *running status* juga dipisahkan oleh *delta-time*, karena dianggap sebagai satu event.

Channel yang digunakan dalam sistem operasi Windows adalah nomor 1-16, tetapi secara umum dianggap sebagai nomor 0-15. Penjelasan mengenai kegunaan tiap channel adalah :

- Channel 0 sampai 8 : *extended melodic track*, 16 nada polyphony.
- Channel 9 : *extended percussian track*, 16 “nada” (bunyi) polyphony.
- Channel 10 dan 11 : tidak digunakan.
- Channel 12 sampai 14 : *base-level melodic track*, 6 nada polyphony.

- Channel 15 : *base-level percussion track*, 3 “nada” (bunyi) polyphony.

Standard MIDI message tersusun atas byte-byte *opcode* diikuti byte-byte data. Secara umum byte opcode disebut sebagai byte status. Byte status ini ditandai dengan memberi nilai 1 pada bit ke-7. Adapun untuk semua byte data mengandung nilai 0 pada bit ke-7, dengan kata lain nilainya berkisar antara 0-127.

MIDI memiliki konsep *logical channel*. Sebagaimana telah dijelaskan, ada 16 channel, yang dinyatakan pada bit 0-3 dari byte status. Sehingga tinggal 3 bit opcode yang digunakan untuk tipe message dengan logical channel. Untuk keperluan tersebut hanya memakai 7 dari 8 (2^3) opcode yang mungkin digunakan. Sedangkan yang tersisa, yaitu yang mengandung nilai 1 pada bit 4-5, digunakan untuk *system message* yang tidak disertai nomor channel. Pada keperluan ini bit 0-3 disediakan untuk spesifikasi lebih jauh mengenai opcode tersebut.

Berikut ini akan dijelaskan spesifikasi untuk setiap byte status yang digunakan dalam file MIDI :

a. Voice Message

Adalah message untuk logical channel yang berhubungan dengan pengolahan suara dalam file MIDI. Spesifikasi message tersebut sebagai berikut :

byte status	arti	byte data
80 - 8F	nada off	2 - 1 byte <i>pitch</i> , diikuti 1 byte <i>velocity</i>
90 - 9F	nada on	2 - 1 byte <i>pitch</i> , diikuti 1 byte <i>velocity</i>
A0 - AF	tekanan kunci	2 - 1 byte <i>pitch</i> , 1 byte <i>pressure A</i>
B0 - BF	parameter	2 - 1 byte no. parameter, 1 byte <i>setting</i>
C0 - CF	program	1 byte pilihan <i>program</i>
D0 - DF	tekanan channel	1 byte <i>channel pressure</i>
E0 - EF	pitch wheel	2 byte dengan nilai 14 bit

Untuk semua message tersebut mendukung penggunaan *running status*. Seperti telah dijelaskan, jika event sebelumnya berisi message dengan type maupun channel yang sama, atau dengan byte status yang sama, maka byte status tidak perlu disertakan lagi.

Pada message nada on, jika *velocity* = 0 maka sama artinya dengan nada off. Dengan demikian akan lebih efisien, karena tidak perlu menyertakan byte status secara berulang-ulang. Sedangkan untuk *velocity normal* = 64.

Byte *pitch* pada byte data menjelaskan tinggi-rendahnya nada, dimana berisi data nada dengan perubahan tiap $\frac{1}{2}$ nada, dengan nada C tengah (C_5 dalam skala 8 oktaf) = 60. Sedangkan untuk perkusi, pitch “nada” memiliki spesifikasi tersendiri sesuai dengan macam perkusi yang bersangkutan, dimana untuk standar sistem operasi Windows spesifikasinya sebagai berikut :

35 acoustic bass drum	51 ride cymbal 1	67 high apogo
36 bass drum 1	52 chimes cymbal	68 low apogo
37 side stick	53 ride bell	69 cabasa
38 acoustic snare	54 tambourine	70 maracas
39 hand clap	55 splash cymbal	71 short whistle
40 electric snare	56 cowbell	72 long whistle
41 low floor tom	57 crash cymbal 2	73 short guiro
42 closed high hat	58 vibraslap	74 long guiro
43 high floor tom	59 ride cymbal 2	75 claves
44 pedal high hat	60 high bongo	76 high wood block
45 low tom	61 low bongo	77 low wood block
46 open high hat	62 mute high conga	78 mute cuica
47 low-mid tom	63 open high conga	79 open cuica
48 high-mid tom	64 low conga	80 mute triangle
49 crash cymbal 1	65 high tymbale	81 open triangle
50 high tom	66 low tymbale	

Pilihan program berkaitan dengan *patch assignment*, dimana untuk standar sistem operasi Windows nomor patch dibagi menjadi 16 kelompok, dengan 8 nilai untuk masing-masing kelompok :

0 - 7	Piano	64 - 71	Reed
8 - 5	Chromatic Percussion	72 - 79	Pipe
16 - 23	Organ	80 - 87	Synth Lead
24 - 31	Guitar	88 - 95	Synth Pad
32 - 39	Bass	96 - 103	Synth Effect
40 - 47	Strings	104 - 111	Ethnic
48 - 55	Ensemble	112 - 119	Percussive
56 - 63	Brass	120 - 127	Sound Effect

Mengenai message parameter, sangat erat hubungannya dengan *controller* instrument. Byte pertama pada byte data memiliki nilai dengan spesifikasi berikut :

0 - 31	<i>continuous controller 0 - 31, most significant byte</i>
32 - 63	<i>continuous controller 0 - 31, least significant byte</i>
64 - 95	<i>on / off switches</i>
96 - 121	tidak dispesifikasikan
122 - 127	<i>channel mode messages</i>

Byte kedua pada byte data mengandung 7 bit setting untuk *controller*. Untuk *switches* byte data 0 = OFF, 127 = ON, dan 1 - 126 tidak didefinisikan. Jika sebuah controller hanya membutuhkan 7 bit resolusi, maka hal tersebut sama dengan menggunakan *most significant* byte. Jika keduanya sama-sama dibutuhkan, maka akan dinyatakan dalam bentuk *most significant* dilanjutkan dengan *least significant*. Dengan sebuah 14 bit controller, diperbolehkan untuk mengirimkan *least significant* byte saja, jika *most significant* tidak perlu diubah.

Nilai untuk *pitch wheel* berkisar antara 0 - 3FFF. Dengan demikian untuk *pitch wheel* normal adalah 2000 (nilai tengah *pitch wheel*), dimana nada tidak mengalami modifikasi.

b. Mode Message

Adalah message yang memiliki byte status B0 - BF, dengan byte data berkisar antara 122 - 127 (lihat message parameter). Byte data ini merupakan penjelasan lebih jauh dari data opcode untuk kelompok message yang mengontrol kombinasi suara dan channel agar bisa dikenali oleh perangkat penerima / pembaca.

Di sini masih terdapat data *basic channel* yang disertakan secara implisit, sehingga perangkat dapat menerima message ini. Arti dari nilai byte data 122 sampai 127 adalah sebagai berikut :

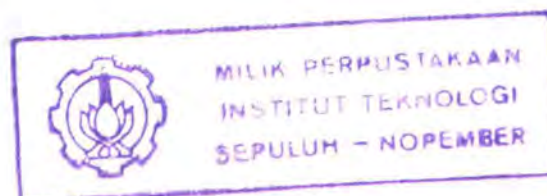
byte pertama	arti	byte kedua
122	local control	0 = local control off, 127 on
123	semua nada off	0
124	omni mode off	0
125	omni mode on	0
126	monophonic mode	jumlah monophonic channel, atau 0 untuk jumlah = suara perangkat penerima
127	polyphonic mode	
124 - 127	juga menyebabkan semua nada off	

Mode setting message mengontrol channel apa, atau berapa suara yang dapat dikenali oleh perangkat penerima. *Omni* menunjukkan kemampuan untuk dapat menerima *voice message* pada semua channel. *Mono* dan *Poly* menunjukkan apakah *multiple voice* diperbolehkan. Permasalahannya adalah bahwa kondisi omni on/off dan kondisi mono/poly bisa berinteraksi satu sama lain. Maka akan terdapat

4 kemungkinan setting, dimana selanjutnya akan disebut sebagai mode dengan penomoran untuk spesifikasi berikut :

- mode 1 - Omni on / Poly - *voice message* diterima pada semua channel dan memperbolehkan polyphoni. Pada dasarnya semua nada dibaca dan dimainkan sampai jumlah suara melebihi batas yang diperbolehkan.
- mode 2 - Omni on / Mono - *monophonic instrument* akan menerima nada-nada untuk dimainkan satu suara pada semua channel.
- mode 3 - Omni off / Poly - *polyphonic instrument* akan menerima *voice message* hanya pada *basic channel*.
- mode 4 - Omni off / Mono - mode yang sering digunakan, untuk mengoperasikan mode ini perangkat penerima dirancang untuk menerima satu suara pada tiap channel. Jumlah channel yang dikenali akan diberikan pada byte data yang kedua, atau sejumlah maksimum kemungkinan suara jika byte tersebut = 0. Keseluruhan channel didefinisikan sebagai rangkaian sequensial, yang diawali oleh *basic channel*.

Pada spesifikasi kondisi tersebut, perangkat penerima boleh mengabaikan mode-mode yang tidak dapat didukung, atau mengubah ke alternatif lain, biasanya menggunakan mode 1. Kebanyakan perangkat penerima dirancang secara *default* berada pada mode 1. Selain itu juga secara umum dikondisikan pada keadaan default dimana perangkat penerima hanya bisa mengenali message nada on / nada off.



c. System Message

Pada byte status F0 - F7 tidak disertai nomor channel. Sehingga byte status tersebut sebenarnya tidak termasuk pada kelompok MIDI event, karena yang termasuk MIDI event adalah kelompok message yang berhubungan dengan *MIDI channel*. Byte F0 - F7 ini digunakan sebagai :

byte status	kegunaan	byte data
F0	system exclusive	<i>variable length</i>
F1	tidak didefinisikan	
F2	song position	2-14 bit <i>value</i> , <i>least significant</i> byte first
F3	song select	1 - song number (nomor lagu)
F4	tidak didefinisikan	
F5	tidak didefinisikan	
F6	tune request	0
F7	EOX (terminator)	0

Song position / song select digunakan untuk mengontrol *sequencer*. Song position dinyatakan dalam bentuk ketukan, yang bisa diinterpretasikan pada setiap 6 pulsa *MIDI clock*. Message ini menentukan apa yang akan dilakukan pada saat menerima message *start real time message* (akan dijelaskan kemudian).

Tune request adalah sebuah perintah pada synthesizer analog untuk menyelaraskan (tune) oscillator masing-masing.

System exclusive message dimaksudkan bagi perancang perangkat pendukung MIDI untuk menyisipkan message-message yang spesifik untuk diaplikasikan pada produk mereka. Byte data berikutnya berkisar antara 0 - 127 untuk setiap byte. Pada system exclusive ini selalu diakhiri dengan F7 sebagai byte *terminator* (pembatas). Byte pertama pada message yang disampaikan, disediakan

untuk *manufacturer's id* (kode untuk perangkat tertentu), yang ditentukan oleh *MIDI standards committee*. Lebih lanjut mengenai system exclusive ini akan dijelaskan kemudian.

Untuk identitas (id) dikelompokkan menurut kode negara produsen, yaitu 0 - 1F hex. untuk produk USA, 20 - 3F hex. untuk produk Eropa, dan 40 - 5F hex. untuk produk Jepang. Berikut ini beberapa *MIDI ID* untuk perangkat dengan merek tertentu :

merek produk	id(hex)	merek produk	id(hex)
Sequential Circuits	1	Bon Tempi	20
Big Briar	2	S.I.E.L.	21
Octave / Plateau	3		
Moog	4	SynteAxe	23
Passport Designs	5		
Lexicon	6		
PAIA	11	Kawai	40
Simmons	12	Roland	41
Gentle Electric	13	Korg	42
Fairlight	14	Yamaha	43

d. Real Time Message

Kelompok terakhir dari byte status adalah F8 - FF. Byte-byte ini disebut sebagai *real time message* karena boleh disisipkan pada semua tempat. Byte status ini bisa dimasukkan diantara byte data pada message yang lain. Secara umum perangkat penerima dirancang untuk dapat menerima dan memproses (atau mengabaikan) message ini, dan melanjutkan pembacaan sisa byte data dari message yang sedang diproses. Real time message tidak mempengaruhi byte *running status* yang sedang berlangsung.

Semua message dalam kelompok ini tidak disertai byte data. Message-message tersebut adalah :

byte status	arti
F8	timing clock
F9	tidak didefinisikan
FA	start
FB	continue
FC	stop
FD	tidak didefinisikan
FE	active sensing
FF	system reset

Message *timing clock* dikirimkan dengan kecepatan 24 clock tiap nada seperempat, dan digunakan untuk perangkat synthesizer, terutama pengolah suara drum (*percussion*).

Start / continue / stop digunakan untuk mengontrol *sequencer* dan pengolah suara drum. Message *continue* menyebabkan perangkat mengambil clock berikutnya untuk ditandai.

Byte *active sensing* dikirimkan setiap 300 milidetik, atau lebih sering jika sedang digunakan. Bentuk ini dirancang untuk mengimplementasikan mekanisme *timeout* bagi perangkat penerima untuk kembali pada state default. Perangkat penerima akan beroperasi secara normal jika mengabaikan message ini, dan mengaktifkan mekanisme *timeout* dari message *active sensing* yang pertama kali. Pada umumnya *active sensing* jarang digunakan.

System reset digunakan untuk inisialisasi sistem seperti pada kondisi awal atau default. Dari spesifikasi ini dapat dikatakan bahwa byte status ini jarang digunakan dan tidak untuk dikirim secara otomatis pada kondisi awal.

2.2.2.2. Sysex Event

<Sysex event> digunakan untuk spesifikasi *MIDI system exclusive message*, yaitu bisa berupa satu unit, atau beberapa paket, atau sebuah *escape* untuk spesifikasi byte-byte perubahan yang akan ditransmisikan. Secara lengkap bentuk normal dari *system exclusive message* yang ada dalam file MIDI adalah sebagai berikut :

F0<length><semua byte yang ditransmisikan setelah F0>

<length> adalah sebagai *variabel-length quantity*. Data ini menjelaskan jumlah byte berikutnya, tidak termasuk F0 maupun panjangnya sendiri. Misalnya message yang akan dikirim adalah F0 43 12 00 07 F7, maka akan disimpan sebagai F0 05 43 12 00 07 F7. Pada byte terakhir selalu disediakan untuk F7, yang digunakan untuk menandai akhir message.

Bentuk lain dari sysex event disediakan untuk event yang tidak mengikutsertakan F0 untuk ditransmisikan. Dalam hal ini digunakan sebuah *escape* untuk memperbolehkan transmisi sesuatu yang tidak umum, seperti *system realtime message*, *song pointer* atau *song select*, *MIDI Time Code*, dan sebagainya. Bentuk *escape* tersebut menggunakan kode F7 sebagai berikut :

F7<length><semua byte yang ditransmisikan>

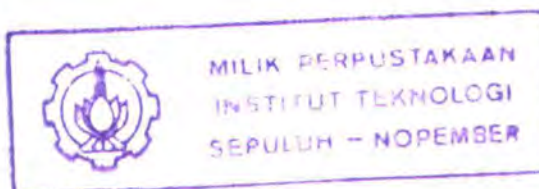
Biasanya perangkat synthesizer mendukung *system exclusive message*, masing-masing dengan spesifikasi khusus, dan ditransmisikan dalam bentuk paket-paket kecil (per paket). Masing-masing paket merupakan bagian dari suatu message. Kode F0 dan F7 digunakan untuk merangkai paket yang dinyatakan dalam beberapa event.

F0 digunakan pada paket pertama dalam suatu rangkaian. Dalam message ini F0 ikut ditransmisikan. Sedangkan F7 digunakan pada paket-paket berikutnya. Pada paket terakhir dalam satu rangkaian, harus diakhiri dengan F7. Hal tersebut sesuai dengan bentuk normal dari *system exclusive message* yang selalu diawali F0, dan diakhiri dengan F7.

Berikut ini adalah contoh bagian dari file MIDI yang mengandung multi paket *system exclusive message* : asumsi paket pertama yang dikirim adalah F0 43 12 00, diikuti dengan delay 200 ketukan, dilanjutkan dengan paket kedua 43 12 00 43 12 00, diikuti dengan delay 100 ketukan, dilanjutkan dengan paket terakhir 43 12 00 F7, maka dalam file MIDI akan dinyatakan sebagai :

```
F0 03 43 12 00
81 48          (delta-time 200 ketukan)
F7 06 43 12 00 43 12 00
64            (delta-time 100 ketukan)
F7 04 43 00 F7
```

Pada saat membaca file MIDI, dan terdapat kode F7 dalam serangkaian multi paket, tanpa diawali F0 pada paket pertama, maka kode F7 tersebut dianggap sebagai *escape*. Dalam hal ini paket terakhir tidak perlu diakhiri dengan F7, dan secara otomatis F7 ikut ditransmisikan.



2.2.2.3. Meta Event

<meta-event> digunakan untuk spesifikasi informasi *non-MIDI*, dengan susunan :

FF<type><length><bytes>

Semua meta event selalu diawali dengan FF, kemudian mempunyai byte yang berisi *event type* (yang selalu lebih kecil dari 128, atau 80 hexadesimal), dan memiliki informasi panjang byte data (*length*) yang dinyatakan dalam bentuk *variable-length quantity*, dan diikuti oleh byte data tersebut. Jika tidak ada data maka *length* = 0. Suatu program mungkin tidak mengenali message yang disampaikan dalam meta-event, maka program tersebut bisa mengabaikannya tanpa mengabaikan byte *length* dari meta-event yang tidak dikenali tersebut.

Baik sysex event maupun meta-event membatalkan semua *running status* yang sedang berlangsung. *Running status* tidak diaplikasikan dan tidak boleh digunakan untuk message ini.

Sebagian meta-event akan dijelaskan di sini, dimana tidak semua program dapat mendukung semua meta-event yang ada.

Pada deskripsi susunan untuk setiap meta-event digunakan bentuk konvensional yang menggambarkan parameter yang terdapat pada event tersebut. Notasi yang mengandung dua huruf kecil seperti dd atau se, berupa data 8 bit (byte), empat huruf kecil yang identik seperti wwww disediakan untuk data 16 bit (word), enam huruf kecil yang identik seperti tttttt disediakan untuk data 24 bit, semuanya dalam bentuk *most significant byte first*.

FF 00 02 wwww**Sequence Number**

Event ini merupakan spesifikasi dari nomor urut track dari serangkaian track yang ada, yang selalu diletakkan di awal track, sebelum semua *delta-time* yang tidak sama dengan 0, dan sebelum semua MIDI event ditransmisikan / dikirimkan. Dalam file MIDI format 2, event ini digunakan untuk identifikasi setiap bagian sehingga sebuah lagu yang terbentuk dari bagian-bagian (track) tersebut akan menggunakan suatu bentuk message yang mengisyaratkan perangkat untuk menghubungi track tersebut. Jika nomor identifikasi tersebut diabaikan, urutan lokasi yang ada pada file digunakan seperti keadaan awal atau default. Pada file MIDI format 0 atau 1, yang hanya terdiri dari satu rangkaian, nomor ini mungkin hanya terdapat pada track awal. Jika sistem mendukung transfer dari beberapa multitrack, maka akan diproses seperti suatu kelompok dari beberapa file format 1, masing-masing dengan *sequence number* (nomor urut) yang berbeda.

Meta-event tipe 01 sampai dengan 0F disediakan untuk bentuk khusus dari *text event*, yang masing-masing berupa spesifikasi dari text event tetapi digunakan untuk keperluan yang berbeda. Berikut ini adalah text event dengan tipe 01 - 07 :

FF 01 len text**Text Event**

Merupakan bentuk umum text event, digunakan untuk meletakkan atau menyisipkan teks dalam file MIDI yang mengandung nilai karakter ASCII.

FF 02 len text**Copyright Notice**

Event ini mengandung *copyright notice* atau informasi pembuat file MIDI tersebut. Biasanya terdiri dari karakter (C), tahun copyright, dan pemilik dari

copyright. Jika beberapa bagian musik dinyatakan dalam satu file MIDI yang sama, maka semua copyright notice sebaiknya diletakkan secara bersamaan pada satu event ini, sehingga berada pada awal file. Event ini sebaiknya sebagai event pertama pada track chunk sebelum semua *delta-time* yang tidak sama dengan 0.

FF 03 len text Sequence / Track Name

Pada track format 0, atau pada track awal format 1, dipakai sebagai nama sequence atau rangkaian track. Selain dari itu digunakan sebagai nama track.

FF 04 len text Instrument Name

Merupakan deskripsi dari perangkat instrumen yang digunakan pada track yang bersangkutan. Biasanya digunakan bersamaan dengan *MIDI Prefix meta-event* sebagai spesifikasi MIDI channel yang digunakan untuk aplikasi perangkat seperti yang dideskripsikan, atau channel yang dispesifikasikan sebagai teks pada event tersebut.

FF 05 len text Lyric

Berisi lirik dari lagu yang sedang dimainkan. Biasanya masing-masing suku kata dinyatakan dalam bentuk *lyric event* yang terpisah, masing-masing pada permulaan *event time* yang mengandung *voice message* yang sesuai dengan lirik tersebut.

FF 06 len text Marker

Umumnya digunakan pada track format 0, atau pada track awal format 1. Maksud dari penamaan rangkaian ini adalah sebagai kata-kata tambahan dalam notasi suatu lagu (seperti *reff*, *chorus*, *bait pertama*, dan sebagainya).

FF 07 len text**Cue Point**

Memberikan deskripsi sesuatu yang terjadi pada sebuah film atau layar video pada saat musik tersebut dimainkan.

FF 20 01 cc**MIDI Channel Prefix**

MIDI chanel (0 - 15) dinyatakan dalam event ini bisa digunakan untuk menghubungkan MIDI channel dengan semua event setelah event ini, termasuk system exclusive dan meta-event. Channel tersebut akan efektif sampai MIDI event normal berikutnya (yang berisi data channel) atau MIDI Channel Prefix meta-event berikutnya. Jika MIDI channel masing-masing satu untuk setiap track, maka message ini hanya digunakan pada file format 0 yang hanya memiliki satu track.

FF 2F 00**End of Track**

Event ini harus selalu digunakan pada semua akhir track pada file MIDI. Sehingga panjang track dapat dispesifikasikan sesuai posisi event ini pada akhir track, yang akan berguna jika terjadi pengulangan maupun penggabungan track.

FF 51 03 tttttt**Set Tempo**

Event ini merupakan indikasi perubahan tempo. Selain itu menyatakan nilai tempo dalam mikrodetik tiap nada seperempat sama dengan 24 kali mikrodetik tiap MIDI clock. Representasi dari tempo pada umumnya adalah sebagai waktu untuk tiap ketukan. Sebaliknya representasi tempo sebagai ketukan untuk tiap satuan waktu dipakai pada bentuk sinkronisasi dengan *time-based synchronizer protocol* sebagaimana yang digunakan pada *SMPTE time code* atau *MIDI time code*. Jumlah akurasi yang disediakan untuk resolusi tempo ini adalah 4 menit untuk satu bagian

pada 120 ketukan tiap menit, akan akurat dalam 500 mikrodetik pada akhir dari satu bagian.

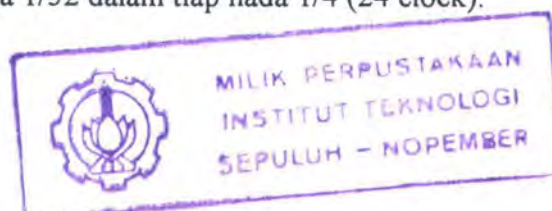
Jadi representasi dari birama 4/4, dengan 120 ketukan per menit adalah FF 51 03 07 A1 20, jika diterjemahkan menjadi 48 MIDI clock tiap detik, 500.000 mikrodetik tiap nada seperempat, 2 nada seperempat tiap detik.

FF 54 05 hr mn se fr ff SMPTE Offset

Event ini diletakkan sebagai desain waktu SMPTE pada tempat dimana track chunk akan dimulai. Event ini juga diletakkan pada awal track, yaitu sebelum semua *delta-time* yang tidak sama dengan 0, dan sebelum semua MIDI event yang dapat dikirimkan. hr (jam) harus dikodekan dalam bentuk format SMPTE, sebagaimana dalam *MIDI Time Code*. Pada file format 1, SMPTE Offset harus dinyatakan beserta *tempo map*, dan tidak berlaku untuk track yang lain. ff merupakan notasi bagian dari frame, dalam 100 bagian tiap frame, sesuai dengan *SMPTE-based track* yang menentukan spesifikasi pembagian frame yang berbeda pada satuan waktu *delta-time*.

FF 58 04 nn dd cc bb Time Signature

Time signature mengekspresikan empat nilai. nn dan dd merepresentasikan *numerator* (pembilang tanda birama) dan *denominator* (penyebut tanda birama). Denominator adalah hasil pangkat negatif dari dua : 2 merepresentasikan nada seperempat, 3 merepresentasikan nada seperdelapan, dan sebagainya. Parameter cc mengekspresikan jumlah MIDI clock pada ketukan metronome. Parameter bb mengekspresikan jumlah notasi nada 1/32 dalam tiap nada 1/4 (24 clock).



Sehingga satu event secara lengkap untuk birama 6/8, dimana ketukan metronome tiap tiga kali nada 1/8, tetapi terdapat 24 clock tiap nada 1/4, 72 clock tiap satu birama, adalah : FF 58 04 06 03 24 08. Dimana birama 6/8 dinyatakan dengan 06 03 ($8 = 2^3$), 36 MIDI clock untuk satu ketukan metronome dinyatakan dengan 24hex. ($3/8$ dibagi $1/4$ dikali 24 clock), dan 8 notasi 1/32 tiap nada 1/4 dinyatakan dengan 08.

FF 59 02 sf mi

Key Signature

Digunakan untuk menentukan nada dasar kunci :

sf = -7 : 7 mol

sf = -1 : 1 mol

sf = 0 : kunci C

sf = 1 : 1 krus

sf = 7 : 7 krus

mi = 0 : kunci mayor

mi = 1 : kunci minor

FF 7F len data

Sequencer Specific

Khusus disediakan untuk *sequencer* dengan spesifikasi yang khusus. Pada byte pertama dari byte data digunakan untuk menyatakan *manufacturer ID*. Sebagaimana MIDI system exclusive, perancang perangkat pendukung MIDI mendefinisikan segala sesuatu yang berhubungan dengan perangkat tersebut dengan memanfaatkan meta-event ini. Sehingga akan memudahkan pemakai dalam menggunakan sequencer yang hanya bisa digunakan untuk format file tertentu.

Pembahasan mengenai file MIDI di atas, digunakan sebagai dasar dalam perancangan perangkat lunak dalam tugas akhir ini. Maka untuk lebih memperjelas

pengenalan terhadap file MIDI, berikut ini ditunjukkan sebuah representasi dari file MIDI format 0 dan format 1.

Delta-Time (desimal)	Kode Event (hex)	Byte Data (desimal)	Keterangan
0	FF 58	04 04 02 24 08	4 byte: birama 4/4, 24 MIDI clock/ketukan metronome, 8 nada $\frac{1}{32}$ / 24 MIDI clock
0	FF 51	03 500000	3 byte: 500.000 mikrodetik / nada $\frac{1}{4}$
0	C0	5	Ch.1 Program 5
0	C1	46	Ch.2 Program 46
0	C2	70	Ch.3 Program 70
0	92	48 96	Ch.3 Nada On C ₄ forte
0	92	60 96	Ch.3 Nada On C ₅ forte
96	91	67 64	Ch.2 Nada On G ₅ mezzo-forte
96	90	76 32	Ch.1 Nada On E ₆ piano
192	82	48 64	Ch.3 Nada Off C ₄ standar
0	82	60 64	Ch.3 Nada Off C ₅ standar
0	81	67 64	Ch.2 Nada Off G ₅ standar
0	80	76 64	Ch.1 Nada Off E ₆ standar
0	FF 2F	00	Akhir track

Berdasarkan isi dari file MIDI di atas, maka bentuk dari file format 0 adalah :

Sebagai awal file adalah header chunk :

4D 54 68 64	Mthd
00 00 00 06	panjang header chunk
00 00	format 0
00 01	1 track
00 60	96 ketukan tiap nada $\frac{1}{4}$

Kemudian track chunk, diawali header kemudian diikuti event-event :

4D 54 72 6B	Mtrk
00 00 00 3B	panjang track chunk (59)

Delta-Time	Event	Keterangan
00	FF 58 04 04 02 18 08	time signature
00	FF 51 03 07 A1 20	tempo
00	C0 05	
00	C1 2E	
00	C2 46	
00	92 30 60	
00	3C 60	<i>running status</i>
60	91 43 40	
60	90 4C 20	
81 40	82 30 40	2 byte <i>delta-time</i>
00	3C 40	<i>running status</i>
00	81 43 40	
00	80 4C 40	
00	FF 2F 00	akhir track

Pada file format 1 representasinya berbeda. Header chunk sebagai berikut :

4D 54 68 64	Mthd
00 00 00 06	panjang header chunk
00 01	format 1
00 04	4 track
00 60	96 ketukan tiap nada $\frac{1}{4}$

Pertama track chunk untuk track time signature / tempo :

4D 54 72 6B	Mtrk
00 00 00 14	panjang track chunk (20)

Delta-Time	Event	Keterangan
00	FF 58 04 04 02 18 08	time signature
00	FF 51 03 07 A1 20	tempo
83 00	FF 2F 00	akhir track

Kemudian track chunk untuk track musik pertama :

4D 54 72 6B	Mtrk
00 00 00 10	panjang track chunk (16)

Delta-Time	Event	Keterangan
00	C0 05	
81 40	90 4C 20	
81 40	4C 00	<i>running status : nada on, vel = 0</i>
00	FF 2F 00	akhir track

Kemudian track chunk untuk track musik kedua :

4D 54 72 6B Mtrk
00 00 00 0F panjang track chunk (15)

Delta-Time	Event	Keterangan
00	C1 2E	
60	91 43 40	
82 20	43 00	<i>running status : nada on, vel = 0</i>
00	FF 2F 00	akhir track

Yang terakhir track chunk untuk track musik ketiga :

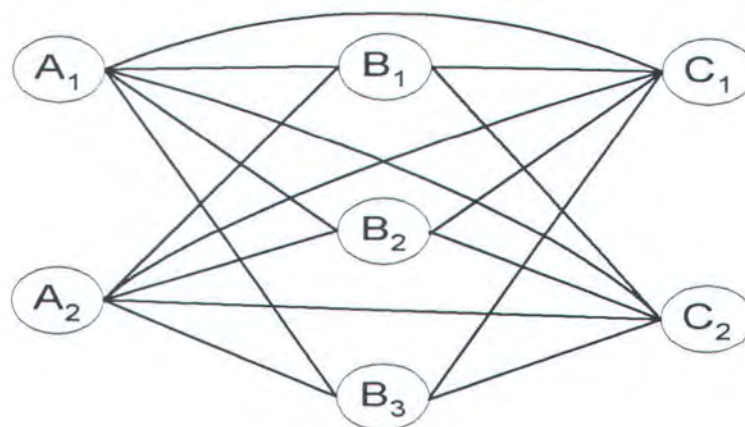
4D 54 72 6B Mtrk
00 00 00 15 panjang track chunk (21)

Delta-Time	Event	Keterangan
00	C2 46	
00	92 30 60	
00	3C 60	<i>running status</i>
83 00	30 00	<i>running status : nada on, vel = 0</i>
00	3C 00	<i>running status : nada on, vel = 0</i>
00	FF 2F 00	akhir track

2.3. Pencarian Jarak Terdekat

Dalam tugas akhir ini, langkah penentuan posisi jari tangan pada fret gitar adalah dengan menggunakan metode pencarian jarak terdekat. Adapun pencarian jarak terdekat yang dimaksud adalah mencari bobot terkecil dari penjumlahan selisih fret semua posisi nada yang membentuk satu rangkaian melodi.

Pada alat musik gitar, satu nada bisa direpresentasikan lebih dari satu cara atau posisi. Sehingga dalam penentuan posisi, harus diusahakan untuk memilih posisi yang bisa dijangkau oleh jari tangan manusia.



Gambar 2.26.

Hubungan antar posisi nada

Metode pencarian jarak terdekat yang digunakan bukan metode yang biasa digunakan pada *weighted graph* (graph dengan bobot) yang umum, seperti misalnya metode *Dijkstra*^[12]. Karena jika pada metode tersebut yang dibutuhkan adalah graph yang dibentuk dari rute perpindahan posisi nada. Padahal yang

^[12] C. L. Liu, *Element of Discrete Mathematics*, second edition, McGraw-Hill Book Company, 1985, pp. 147-149

dibutuhkan pada proses penentuan posisi pada *fretboard* adalah graph yang dibentuk oleh keterhubungan antar semua posisi nada yang ada.

Pada gambar 2.26 ditunjukkan contoh graph yang dibentuk oleh keterhubungan posisi-posisi nada. Misalkan terdapat tiga nada yang berurutan atau yang dibunyikan secara bersamaan, yaitu A, B, dan C. Masing-masing nada mempunyai jumlah kemungkinan posisi yang berbeda : A memiliki relasi 2 posisi pada fretboard yaitu A_1 dan A_2 , B memiliki relasi 3 posisi pada fretboard B_1 , B_2 , dan B_3 , dan C memiliki relasi 2 posisi yaitu C_1 dan C_2 . Maka akan terjadi kemungkinan urutan posisi dengan persamaan :

$$\Sigma \text{kemungkinan} = \Sigma \text{posisi A} \times \Sigma \text{posisi B} \times \Sigma \text{posisi C} \quad (2.1)$$

Dari persamaan tersebut didapatkan jumlah kemungkinan yang terjadi adalah $2 \times 3 \times 2 = 12$ kemungkinan, yaitu $A_1-B_1-C_1$, $A_1-B_2-C_1$, $A_1-B_3-C_1$, $A_1-B_1-C_2$, $A_1-B_2-C_2$, $A_1-B_3-C_2$, $A_2-B_1-C_1$, $A_2-B_2-C_1$, $A_2-B_3-C_1$, $A_2-B_1-C_2$, $A_2-B_2-C_2$, dan $A_2-B_3-C_2$.

Jika pada *weighted graph* yang dibentuk oleh rute perpindahan posisi nada maka *edge* A_1-C_1 , A_1-C_2 , A_2-C_1 , dan A_2-C_2 tidak disertakan. Sehingga jika menggunakan metode *Dijkstra*, jika pada *vertex* B_1 diketahui bahwa bobot yang didapatkan dari *edge* A_1-B_1 lebih kecil dari bobot *edge* A_2-B_1 , maka bobot *vertex* B_1 sama dengan bobot yang didapatkan dari *edge* A_2-B_1 . Jika dilanjutkan pada salah satu rute berikutnya, misalkan pada *vertex* C_1 , maka rute $A_2-B_1-C_1$ sudah diabaikan, padahal masih ada kemungkinan bahwa bobot *edge* A_2-C_1 lebih kecil bobot *edge* A_1-C_1 .

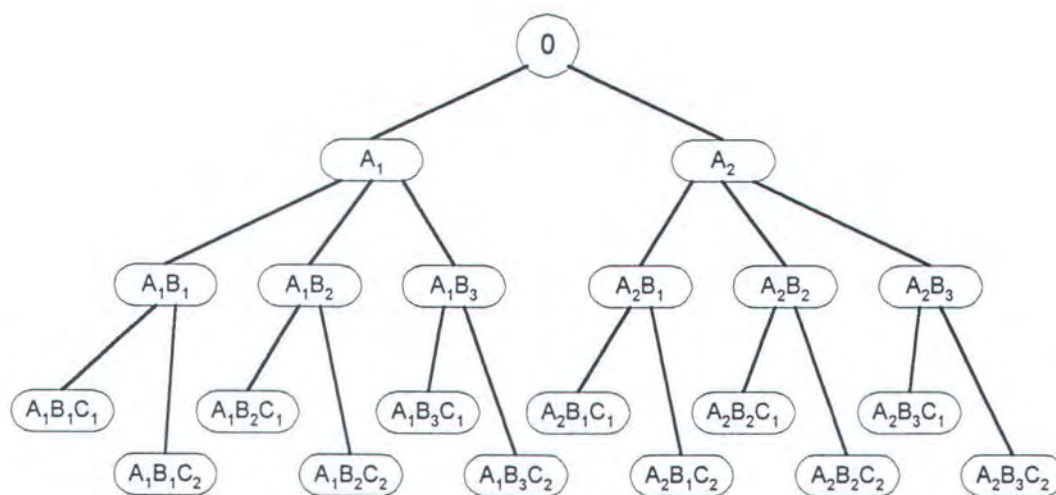
Berdasarkan kemungkinan sebagaimana pada persamaan (2.1) akan dipilih salah satu yang mempunyai jarak paling berdekatan. Yaitu berdasarkan jumlah bobot *edge* yang diperoleh menghubungkan ketiga posisi nada. Bobot tersebut adalah selisih dari posisi fret. Sebagai contoh kita ambil $A_1-B_1-C_1$, maka jaraknya bisa didefinisikan sebagai persamaan :

$$\overline{A_1B_1C_1} = \overline{A_1B_1} + \overline{A_1C_1} + \overline{B_1C_1} \quad (2.2)$$

Contoh diatas menunjukkan bahwa jika jumlah nada adalah N maka jumlah *edge* E yang menghubungkan semua posisi pada satu kemungkinan adalah :

$$E = \frac{N(N-1)}{2} \quad (2.3)$$

Keterhubungan posisi nada A, B dan C dengan bobot yang merupakan selisih posisi fret, dapat diimplementasikan sebagai pohon (*tree*) sebagaimana ditunjukkan pada gambar 2.27 berikut.



Gambar 2.27.

Pohon kombinasi posisi nada



State 0 dianggap sebagai keadaan awal, dengan bobot 0. Untuk memilih salah satu dari 12 kemungkinan yang ada, maka bisa digunakan suatu metode yang bisa mengunjungi semua daun (*leaf*) dengan membawa suatu nilai bobot sebagai pembanding. Alternatif yang dapat dipilih untuk keperluan tersebut adalah metode *Branch and Bound* atau dengan metode *Apha-Beta Pruning* dengan *minimizing level*^[13]. Dalam hal ini mencari bobot terkecil (*least-cost*) dari daun yang ada.

Diawali dari state 0 menuju ke state $A_1B_1C_1$ melalui A_1 dan A_1B_1 . Kemudian bobot $A_1B_1C_1$ dibawa ke state $A_1B_1C_2$ untuk dibandingkan melalui A_1B_1 . Hasil perbandingan yang ada pada state A_1B_1 dibawa ke state $A_1B_2C_1$ melalui A_1 dan A_1B_2 . Dengan langkah yang sama pada state A_1B_1 , hasil perbandingan dengan $A_1B_2C_1$ dibawa ke state $A_1B_2C_2$ melalui state A_1B_2 . Demikian selanjutnya sampai dengan state $A_2B_3C_2$.

Pada implementasi yang sebenarnya, dimana bobot yang dimaksud adalah jumlah jarak antara dari semua posisi yang membentuk satu daun, pada tiap state yang bukan daun akan membawa data posisi dari semua state yang berada di atasnya, selain data posisinya sendiri. Bobot yang didapatkan dari selisih posisi, akan diperoleh jika telah mencapai daun. Dari gambaran tersebut maka didapatkan suatu algoritma yang bersifat rekursif untuk pencarian jarak terdekat atau bobot yang terkecil, yaitu sebagai berikut :

^[13] Patrick Henry Winston, **Artificial Intelligence**, second edition, Addison-Wesley Publishing Company, 1984, pp. 87-132

1. Jika A salah satu posisi awal (bisa lebih dari satu) maka total jarak = 0, posisi *parent* tidak diperhitungkan. Jika bukan posisi awal maka jarak = jarak awal + $\sum^n |\text{posisi } \textit{parent}[n] - \text{posisi } A|$, dimana n adalah jumlah *parent*.
2. Jika A bukan daun, maka dilanjutkan untuk semua state dibawahnya dengan langkah 1, dengan menyertakan jarak sebagai jarak awal serta posisi A dan semua posisi *parent*-nya sebagai posisi *parent*.
3. Jika A adalah daun maka jarak adalah bobot dari state dengan urutan posisi-posisi *parent* sampai dengan A. Bobot tersebut dibandingkan dengan bobot pembanding, jika bobot A lebih kecil maka untuk selanjutnya dipakai sebagai bobot pembanding.
4. Bobot pembanding yang terakhir merupakan bobot yang terkecil atau jarak yang terpendek.

Pada bentuk algoritma tersebut, untuk posisi nada dengan jumlah n, pada rekursi ke-(n-1) akan menyertakan posisi *parent* sebanyak n-1. Hal ini berarti membutuhkan penampung variabel yang sangat besar untuk data posisi *parent*, dan stack memory yang besar untuk proses rekursi. Maka untuk mengatasi hal tersebut, data yang akan dikelompokkan dalam batasan interval yang telah ditentukan. Dalam tugas akhir ini digunakan batasan interval satu oktaf. Batasan ini diambil berdasarkan pola nada pada fret gitar. Satu oktaf nada pada fret gitar bisa direpresentasikan tidak lebih dari 5 fret. Sehingga jumlah rekursi maksimal adalah 11 kali, karena jumlah nada dalam satu oktaf adalah 12 nada. Misalkan suatu melodi terdiri dari rangkaian 100 nada yang berurutan, dengan interval satu oktaf,

maka hanya membutuhkan satu kali proses. Jika 100 nada tersebut berelasi dengan 7 nada dalam satu oktaf, maka hanya membutuhkan proses rekursi sebanyak 6 kali. Setelah didapatkan posisi untuk 7 nada tersebut, maka selanjutnya merelasikan posisi-posisi tersebut dengan rangkaian nada pada melodi yang diproses.

Sebagai asumsi suatu melodi terdiri dari 5 nada yang berurutan dengan interval lebih dari satu oktaf, maka algoritma di atas akan lebih optimal jika digunakan dengan membagi menjadi interval-interval satu oktaf, daripada jika digunakan untuk keseluruhan melodi. Misalnya urutan nada-nada dari melodi yang dipakai sebagai asumsi adalah $F\#_3$, G_3 , $A\#_3$, B_3 , dan D_7 (kebetulan pada asumsi ini tidak ada nada yang sama). Masing-masing nada memiliki relasi data posisi sebagai berikut (tanpa menyertakan posisi senar) :

$F\#_3$: 2
G_3	: 3
$A\#_3$: 1, 6
B_3	: 2, 7
D_7	: 22

Jika dilakukan proses perhitungan untuk semua nada, maka kemungkinannya adalah:

- $$|2-3| + |2-1|+|3-1| + |2-2|+|3-2|+|1-2| + |2-22|+|3-22|+|1-22|+|2-22| = 1+1+2+0+1+1+20+19+21+20 = 86$$
- $$|2-3| + |2-1|+|3-1| + |2-7|+|3-7|+|1-7| + |2-22|+|3-22|+|1-22|+|7-22| = 1+1+2+5+4+6+20+19+21+20+15 = 114$$
- $$|2-3| + |2-6|+|3-6| + |2-2|+|3-2|+|1-2| + |2-22|+|3-22|+|6-22|+|2-22| = 1+4+3+0+1+1+20+19+16+20 = 85$$

$$d. |2-3| + |2-6|+|3-6| + |2-7|+|3-7|+|1-7| + |2-22|+|3-22|+|6-22|+|7-22| = 1+4+3+5+4+6+20+19+21+16+15 = 114$$

Dari perhitungan tersebut didapatkan posisi optimalnya adalah 2-3-6-2-22. Posisi yang dipilih akan berbeda jika menggunakan batasan interval. Asumsi kita gunakan interval satu oktaf, sehingga terdapat interval pertama $F\#_3$, G_3 , $A\#_3$ dan B_3 , dan interval kedua D_7 , maka perhitungannya adalah sebagai berikut :

Untuk interval pertama :

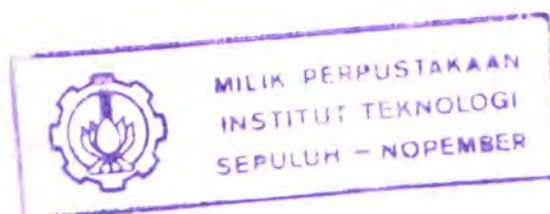
$$a. |2-3| + |2-1|+|3-1| + |2-2|+|3-2|+|1-2| = 1+1+2+0+1+1 = 5$$

$$b. |2-3| + |2-1|+|3-1| + |2-7|+|3-7|+|1-7| = 1+1+2+5+4+6 = 20$$

$$c. |2-3| + |2-6|+|3-6| + |2-2|+|3-2|+|1-2| = 1+4+3+0+1+1 = 10$$

$$d. |2-3| + |2-6|+|3-6| + |2-7|+|3-7|+|1-7| = 1+4+3+5+4+6 = 23$$

Sedangkan untuk interval kedua hanya ada 1 posisi sehingga tidak perlu dihitung. Jika lebih dari satu diambilkan posisi terakhir yang telah ditentukan pada interval pertama. Sehingga dengan metode ini kita dapatkan posisi optimalnya adalah 2-3-1-2-22. Dari hasil tersebut maka bisa dilihat bahwa metode kedua lebih optimal dari metode pertama, karena pada metode pertama terjadi perpindahan yang jauh yaitu 3-6-2, sedangkan pada metode yang kedua didapatkan 3-1-2. Hal tersebut menunjukkan bahwa dalam pencarian posisi pada fretboard harus ditentukan batasan interval yang digunakan untuk membagi rangkaian nada dalam suatu melodi yang memiliki interval nada yang besar, untuk mendapatkan penempatan posisi yang optimal.



BAB III

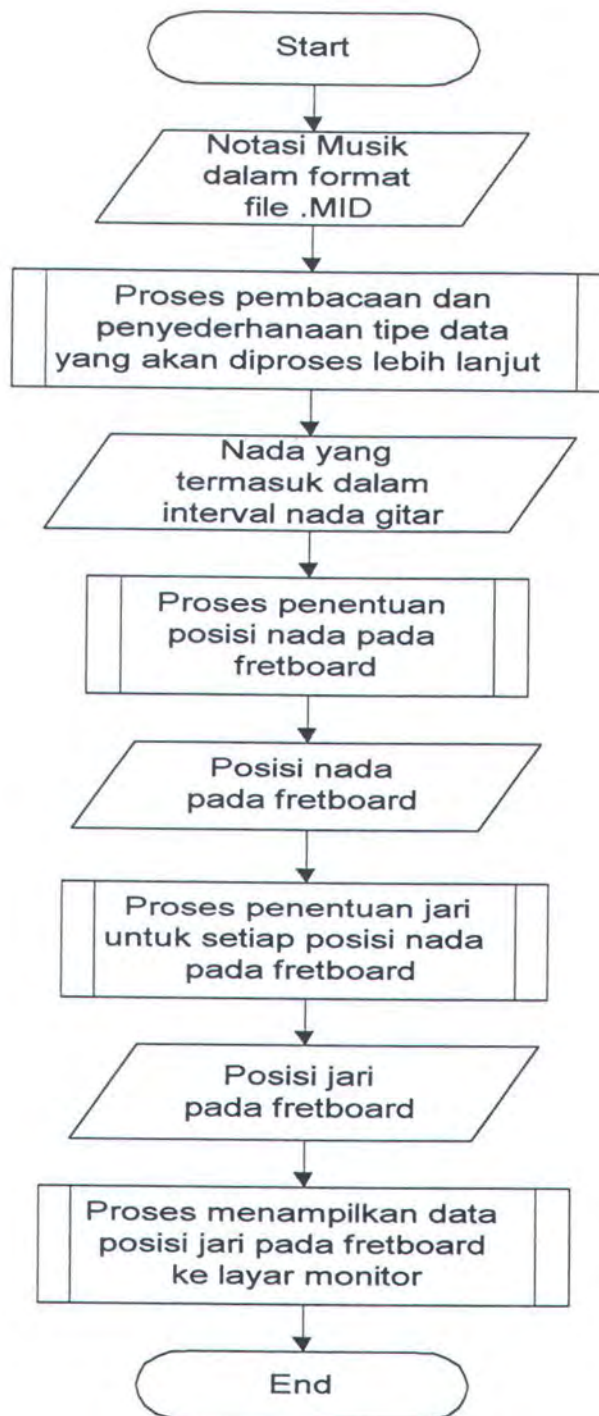
PERENCANAAN PROGRAM

Di dalam pembuatan sebuah program, perencanaan sistem dan struktur data merupakan hal yang penting. Karena banyak cara dan metode yang dapat dipakai dalam pemecahan suatu masalah, maka perlu dipilih cara dan metode terbaik berdasarkan kondisi-kondisi yang ada, misalnya tipe dan karakteristik data masukan dan hasil yang diinginkan, kapasitas penyimpanan, dan kondisi lain yang mempengaruhi. Demikian halnya dengan perencanaan struktur data, karena suatu program pada dasarnya melakukan proses transformasi bentuk data, dari bentuk data masukan menjadi bentuk data hasil yang diinginkan.

Di samping itu, perencanaan sistem dan struktur data program juga harus memperhatikan bahasa pemrograman dan sistem operasi yang digunakan untuk mendukung aplikasi tersebut. Sehingga mudah diimplementasikan dengan bahasa pemrograman dan sistem operasi yang digunakan.

3.1. Perencanaan Sistem

Program penerjemah notasi musik menjadi tampilan posisi jari pada fret gitar ini terdiri dari pemrosesan yang bertahap, dimulai dari pemasukan data notasi musik dalam format file .MID, sampai dihasilkan data posisi jari pada fret gitar yang siap ditampilkan. Perencanaan sistem seperti yang ditunjukkan pada gambar 4.1.



Gambar 3.1.

Diagram sistem penerjemahan notasi musik menjadi tampilan posisi jari pada fret gitar

Diagram perencanaan sistem penerjemahan notasi musik menjadi tampilan posisi jari pada fret gitar tersebut terdiri dari dua komponen, yaitu komponen data (masukan/hasil) dan komponen pemrosesan.

3.1.1. Komponen Data

Komponen data dalam perencanaan sistem program ini terdiri dari 4 komponen utama yaitu :

a. Notasi Musik dalam Format File .MID sebagai Masukan

Data notasi musik ini merupakan data dalam bentuk file .MID yang sudah siap diproses. Data masukan ini dibatasi untuk file .MID format 0 atau format 1, dan bukan dalam format SMPTE (Society for Motion Picture and Television Engineers), dimana hal tersebut dapat diketahui dari header file .MID.

Penjelasan lebih lanjut pada sub bab 3.2.1 halaman 67.

b. Nada yang Termasuk dalam Interval Nada Gitar

Data nada yang termasuk dalam interval nada gitar diperoleh dari proses penyederhanaan dari data masukan. Data ini memuat nada yang berada dalam interval nada gitar, dan maksimal nada yang dibunyikan pada satu saat adalah 6 nada. Disamping itu data ini juga mengabaikan semua efek suara yang terdapat pada data masukan.

Penjelasan lebih lanjut pada sub bab 3.2.2 halaman 68.

c. Posisi Nada pada Fretboard

Data posisi nada pada *fretboard* merupakan hasil pengolahan data nada yang termasuk dalam interval nada gitar. Pengolahan yang dimaksud adalah pencarian jarak terdekat. Sebagaimana telah dijelaskan sebelumnya bahwa sebuah data nada memiliki relasi dengan satu atau lebih data posisi pada fret gitar, maka

sebelum diproses data nada tersebut sebelumnya telah drelasikan dengan data posisi yang sudah dideklarasikan sebagai nilai konstanta.

Penjelasan lebih lanjut pada sub bab 3.2.3 halaman 69.

d. Posisi Jari pada Fretboard

Data ini adalah data posisi nada pada *fretboard* yang telah dilengkapi dengan data nomor jari (jari tangan kiri) yang menempati posisi nada yang dimaksud. Ada kemungkinan terdapat beberapa posisi nada yang diabaikan karena posisi jari yang tidak memungkinkan. Hal tersebut berlaku untuk nada-nada yang dibunyikan bersamaan pada satu waktu.

Penjelasan lebih lanjut pada sub bab 3.2.4 halaman 70.

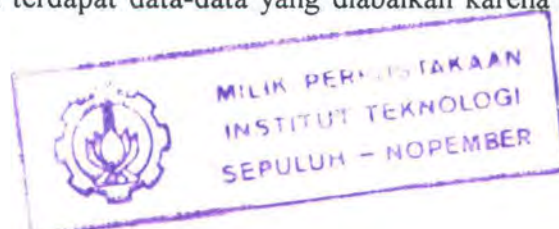
3.1.2. Komponen Pemrosesan

Secara garis besar, komponen pemrosesan terbagi menjadi empat komponen, yaitu :

a. Pembacaan dan Penyederhanaan File .MID

Proses ini diawali dengan membaca *MIDI file header*. Dari data header dapat diketahui apakah file MIDI yang dibaca memiliki format 0, 1, atau 2, maupun format SMPTE atau standar.

Proses selanjutnya adalah membaca data track, dimana tersimpan data notasi musik, instrumen musik, maupun efek suara. Telah dijelaskan sebelumnya, bahwa program ini hanya mengambil data tinggi rendahnya nada (*pitch*) yang ada pada file MIDI, dengan demikian terdapat data-data yang diabaikan karena tidak



dibutuhkan pada proses berikutnya. Data yang merupakan penyederhanaan dari file MIDI tersebut disimpan ke dalam format tertentu, untuk selanjutnya dianggap sebagai file *temporary*.

Penjelasan lebih lanjut pada sub bab 4.3 halaman 79.

b. Penentuan Posisi Nada pada FretBoard

Proses ini bertujuan untuk merelasikan satu nada dengan satu posisi pada *fretboard*. Metode yang digunakan dalam proses ini adalah metode pencarian jarak terdekat (*shortest path*).

Data dari file *temporary* hasil proses penyederhanaan file MIDI direlasikan dengan data posisi tiap nada pada *fretboard* yang telah dideklarasikan sebagai konstanta *array*, dimana pada gitar yang memiliki 24 fret, tiap nada memiliki satu sampai dengan maksimal enam posisi pada *fretboard*. Setelah ditemukan posisi yang paling optimal, data tersebut ditambahkan pada file *temporary* untuk diolah pada proses berikutnya.

Penjelasan lebih lanjut pada sub bab 4.4 halaman 82.

c. Penentuan Jari untuk Posisi Nada pada Fretboard

Penentuan jari untuk posisi nada pada *fretboard* adalah proses penambahan data yang terakhir pada file hasil keluaran, dimana satu posisi nada pada *fretboard* akan direlasikan dengan notasi jari (tangan kiri) yang digunakan untuk menekan senar gitar pada posisi fret yang dimaksud.

Penjelasan lebih lanjut pada sub bab 4.5 halaman 88.

d. Menampilkan Posisi Jari pada Fretboard ke Layar Monitor

Proses ini berkaitan dengan pengolahan antar muka visual yang akan menampilkan data file hasil keluaran sebagai tampilan posisi jari (notasi tangan kiri) pada *fretboard* ke layar monitor. Hasil implementasi proses ini dirancang agar pemakai dapat mengikuti notasi gitar yang ditampilkan. Oleh sebab itu tempo permainan gitar yang ditampilkan bisa diatur sesuai dengan keinginan pemakai.

Penjelasan lebih lanjut pada sub bab 4.6 halaman 95.

3.2. Perencanaan Struktur Data

Perencanaan struktur data adalah komponen yang penting dalam perancangan program. Hal yang berkaitan dengan perencanaan tersebut adalah bentuk data masukan, data keluaran, serta variabel-variabel lain yang dibutuhkan pada pemrosesan data.

3.2.1. Notasi Musik dalam Format File MIDI

Data masukan adalah data notasi musik yang disimpan sebagai file MIDI yang memiliki format seperti yang telah dijelaskan pada Bab 2, yaitu terdiri dari *header chunks* dan *track chunks*. Data notasi musik disimpan sebagai rangkaian *MTrk (MIDI track) event* pada *track chunks*, dimana kegunaan, panjang data maupun format pembacaan data tersebut tergantung dengan tipe event.

Karena setiap *MTrk event* memiliki panjang data yang berbeda, maka program yang dibuat akan membaca data per byte, kemudian data tersebut akan

diproses sesuai dengan tipe event. Untuk memudahkan pembacaan file MIDI, terutama pada pengenalan event, maka tipe event yang ada dideklarasikan sebagai nilai konstanta.

Struktur file MIDI yang dibaca kemudian disederhanakan, karena proses penerjemahan ini hanya membutuhkan data *division*, *meta event set tempo*, *meta event time signature*, *midi event note on*, dan *delta-time*.

Penjelasan lebih lanjut pada sub bab 4.2.1 halaman 72.

3.2.2. Nada dalam Interval Nada Gitar

Sebagai hasil proses penyederhanaan file MIDI, data-data yang akan diproses lebih lanjut, yaitu data nada yang masuk dalam interval nada gitar, ditampung sebagai file *temporary* dengan struktur data tersendiri. File *temporary* ini direncanakan untuk dapat memudahkan proses selanjutnya. Adapun format pembacaan delta-time pada file ini sama dengan format pembacaan delta-time pada file MIDI, yaitu sebagai *variable length quantity*. Format variabel nada sama dengan format variabel *pitch* pada file MIDI, dengan nilai berkisar antara 40 (E3) sampai 88 (E7). Sedang data di luar interval tersebut akan dikenali sebagai *event kosong*, sehingga pada proses berikutnya event tersebut diabaikan dan tidak akan mengalami perubahan.

Penjelasan lebih lanjut pada sub bab 4.2.2 halaman 74.

3.2.3. Posisi Nada pada Fretboard

Data posisi nada pada *fretboard* adalah berupa data byte yang akan ditambahkan pada struktur data file hasil penyederhanaan file MIDI. Penambahan data posisi dilakukan pada setiap data nada yang bisa diproses. Sehingga pada setiap byte data nada akan diikuti satu byte data posisi. Sedangkan untuk nada-nada yang tidak bisa diproses lebih lanjut, akan diubah menjadi *event kosong*.

Untuk menentukan data posisi yang akan ditambahkan pada struktur tersebut, maka semua posisi tiap nada yang ada pada *fretboard* harus diketahui, untuk kemudian diolah dengan menggunakan metode pencarian jarak terdekat (*shortest path*). Untuk itu 49 nada yang ada dalam interval nada gitar direlasikan dengan data posisi masing-masing, hal tersebut bisa direpresentasikan sebagai nilai konstanta array dua dimensi yaitu berdasarkan jumlah nada pada interval nada gitar dengan 24 fret (49 nada) dan posisi-posisi untuk tiap nada pada *fretboard* (maksimal 5 posisi).

Berikutnya adalah proses *shortest path*, yang akan memanfaatkan variabel-variabel *array* yang berisi data posisi. Proses ini akan mengolah maksimal 6 nada pada satu kali proses, hal ini disesuaikan dengan besarnya interval yang dipilih dan jumlah senar gitar (nada yang bisa dibunyikan pada saat yang sama).

Penjelasan lebih lanjut pada sub bab 4.2.3 halaman 75.

3.2.4. Posisi Jari pada Fretboard

Data terakhir yang ditambahkan pada file hasil keluaran adalah posisi jari pada *fretboard*. Dimana yang digunakan adalah notasi jari tangan kiri, yaitu jari no.1 (telunjuk), no.2 (jari tengah), no.3 (jari manis), dan no.4 (kelingking). Data nomor jari tersebut ditambahkan pada file yang telah memuat data nada dan posisi.

Hasil proses konversi yang terakhir ini juga memungkinkan terjadinya pengabaian data nada yang tidak bisa direpresentasikan. Yaitu nada yang dibunyikan bersamaan yang membutuhkan lebih dari empat jari untuk merepresentasikannya. Seperti halnya data posisi, data jari juga berupa data byte yang disertakan pada setiap nada.

Penjelasan lebih lanjut pada sub bab 4.2.4 halaman 78.

BAB IV

PEMBUATAN PROGRAM

Dalam pembahasan pembuatan program ini dijelaskan mengenai gambaran umum perangkat lunak dan modul-modul program beserta algoritma proses untuk tiap modul, sebagai implementasi dari perencanaan program yang telah dibahas pada Bab III.

4.1. Umum

Seperti telah dijelaskan dalam Bab III bahwa perangkat lunak penerjemah notasi musik menjadi posisi jari pada fret gitar ini dirancang untuk bisa mengkonversikan file data masukan menjadi file data hasil keluaran yang siap ditampilkan. Adapun hal tersebut tidak terlepas dari pengolahan komponen data melalui tahapan-tahapan yang ada dalam komponen pemrosesan.

Pada tahap perencanaan telah dibahas mengenai struktur data yang akan digunakan dalam pembuatan program ini. Maka sebagai implementasi dari tahap perencanaan tersebut, program ini dibagi menjadi 4 modul utama, sesuai tahapan pemrosesan. Modul-modul tersebut adalah :

- a. Modul untuk membaca dan mengkonversi file MIDI
- b. Modul pemilihan posisi nada pada fretboard
- c. Modul penentuan jari untuk tiap posisi pada fretboard
- d. Modul pengolah tampilan hasil keluaran

4.2. Struktur Data

Pembahasan berikut ini mengenai struktur data yang berkaitan dengan file-file yang digunakan sesuai dengan format penyimpanannya serta data-data penunjang yang dibutuhkan oleh komponen pemrosesan. Adapun struktur data yang akan diuraikan berikut ini merupakan implementasi dari tahap perencanaan struktur data yang telah diuraikan pada Bab III. Bentuk struktur data disesuaikan dengan format bahasa pemrograman Delphi atau Pascal yang digunakan dalam pembuatan program dalam tugas akhir ini.

4.2.1. Notasi Musik dalam Format File MIDI

Seperti telah dijelaskan pada Bab II, bahwa struktur file MIDI terdiri dari *header chunks* dengan struktur data :

<chunk type><length><format><ntrks><division>

ChunkType	: longint;	{ <i>tipe chunk</i> }
Length	: longint;	{ <i>panjang byte chunk</i> }
Format	: word;	{ <i>format file MIDI</i> }
NTrks	: word;	{ <i>jumlah track</i> }
Division	: word;	{ <i>pembagi delta time</i> }

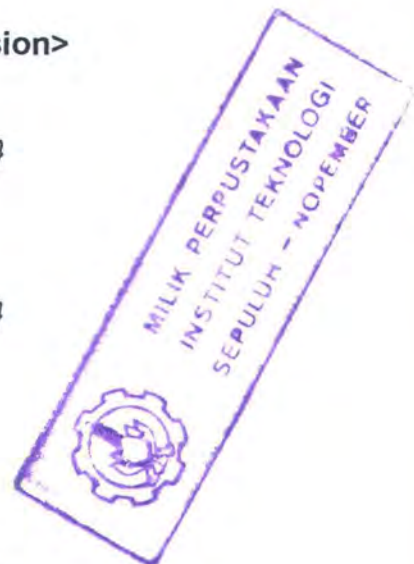
dan *track chunks* dengan struktur data :

<chunk type><length><MTrk event>+

ChunkType	: longint;	{ <i>tipe chunk</i> }
Length	: longint;	{ <i>panjang byte chunk</i> }

Data notasi musik disimpan sebagai rangkaian *MTrk* (*MIDI track*) *event* pada track chunks. Setiap *MTrk event* memiliki struktur data sebagai berikut :

<delta-time><event type><length><data>+



DeltaTime : (1-4 bytes); {variable-length quantity}
 EventType : byte; {tipe event}
 Length : byte; {panjang byte event / jumlah data}
 Data : byte; {data event}

Pada struktur *Mtrk event* tersebut jumlah variabel *data* ditentukan oleh variabel *length*. Sedangkan format pembacaan tiap byte data tergantung dari tipe event.

Untuk memudahkan pembacaan file MIDI, tipe event yang ada dideklarasikan sebagai nilai konstanta :

const

{chunk type}

MThdChunk = \$4D546864; {header chunk type}

MTrkChunk = \$4D54726B; {track chunk type}

{MIDI event type}

ME_NoteOff = \$80;

ME_NoteOn = \$90;

ME_KeyPressure = \$A0;

ME_Parameter = \$B0;

ME_Program = \$C0;

ME_ChannelPressure = \$D0;

ME_PitchWheel = \$E0;

{system exclusive event type}

ME_SysexEvent = \$F0;

ME_SongPosition = \$F2;

ME_SongSelect = \$F3;

ME_TuneRequest = \$F6;

ME_EOX = \$F7;

ME_MetaEvent = \$FF;

{meta event type}

META_SequenceNo = \$00;

META_Text = \$01;

META_CopyrightNtc = \$02;

META_TrackName = \$03;

META_InstrumentName = \$04;

META_Lyric = \$05;

META_Marker = \$06;

META_CuePoint = \$07;

META_ChannelPrefix	= \$20;
META_EndOfTrack	= \$2F;
META_SetTempo	= \$51;
META_SMPTEOffset	= \$54;
META_TimeSignature	= \$58;
META_KeySignature	= \$59;
META_Specific	= \$7F;

Struktur file MIDI tersebut kemudian disederhanakan, yaitu dengan hanya memanfaatkan data *division*, *meta event set tempo*, *meta event time signature*, *midi event note on*, dan *delta-time*.

4.2.2. Nada dalam Interval Nada Gitar

Sebagai hasil proses penyederhanaan file MIDI, data-data yang akan diproses lebih lanjut, yaitu data nada yang masuk dalam interval nada gitar, ditampung sebagai file temporary dengan struktur data sebagai berikut :

<division><time signature><set tempo><event>+...<end>

Division : word;
 TimeSignature : (5 bytes);
 SetTempo : (4 bytes);

Bentuk struktur *TimeSignature* adalah :

D0 nn dd cc bb

D0 : nilai konstan;
 nn : *nominator*, pembilang tanda birama
 dd : *denominator*, 2^{dd} = penyebut tanda birama
 cc : MIDI *clocks* untuk tiap ketukan metronome
 bb : jumlah nada 1/32 tiap 24 MIDI clock (tiap nada 1/4)

Bentuk struktur *SetTempo* adalah :

C0 tt tt tt

C0 : nilai konstan

tt tt tt : 24 bit nilai tempo untuk tiap nada 1/4 dalam mikrodetik

Sedangkan untuk *event* memiliki struktur sebagai berikut :

<nothing | gitar note><delta-time>

DeltaTime : **(1-4 bytes); {variable-length quantity}**

Variabel *nothing* adalah data-data dari file MIDI yang diabaikan, berisi nilai konstan **A0**. Variabel *end* menandai akhir file, berisi nilai konstan **B0**. Sedangkan *gitar note* memiliki struktur :

FF <length><note>+

FF : nilai konstan

Length : **byte;** {panjang byte gitar note / jumlah nada}

Note : **byte;** {data nada}

Nilai variable *length* berkisar antara 1 sampai 6, yaitu jumlah nada yang dibunyikan bersamaan pada satu waktu. Format variabel *note* sama dengan format variabel *pitch* pada file MIDI, dengan nilai berkisar antara 40 (E3) sampai 88 (E7).

File temporary dengan format tersebut akan disimpan sebagai file GT1.TMP.

4.2.3. Posisi Nada pada Fretboard

Posisi nada pada fretboard merupakan pelengkap dari struktur data file hasil penyederhanaan file MIDI atau file GT1.TMP. Penambahan data posisi dilakukan pada setiap data nada dalam struktur *gitar note* sehingga menjadi :

FF <length><note+position>+

FF : nilai konstan;

Length : **byte;** {jumlah nada}

Note : **byte;** {data nada}

Position : **byte**; {data posisi}

Untuk menentukan data posisi yang akan ditambahkan pada struktur tersebut, maka semua posisi tiap nada yang ada pada fretboard harus diketahui, untuk kemudian diolah dengan menggunakan metode pencarian jarak terdekat (*shortest path*). Untuk itu 49 nada yang ada dalam interval nada gitar direlasikan dengan data posisi masing-masing, dan dideklarasikan sebagai konstanta array dua dimensi sebagai berikut :

const

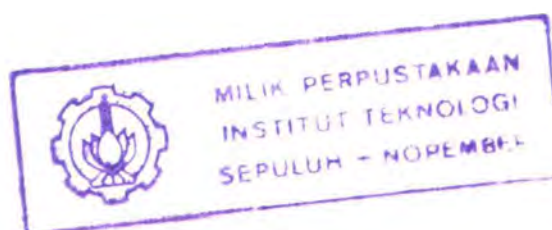
ArrayNada : array [0..48,0..6] of byte =

(1,	6,	0,	0,	0,	0,	0),	{posisi nada E3}
(1,	16,	0,	0,	0,	0,	0),	{posisi nada F3}
(1,	26,	0,	0,	0,	0,	0),	{posisi nada F#3}
(1,	36,	0,	0,	0,	0,	0),	{posisi nada G3}
(1,	46,	0,	0,	0,	0,	0),	{posisi nada G#3}
(2,	56,	5,	0,	0,	0,	0),	{posisi nada A3}
(2,	66,	15,	0,	0,	0,	0),	{posisi nada A#3}
(2,	76,	25,	0,	0,	0,	0),	{posisi nada B3}
(2,	86,	35,	0,	0,	0,	0),	{posisi nada C4}
(2,	96,	45,	0,	0,	0,	0),	{posisi nada C#4}
(3,	106,	55,	4,	0,	0,	0),	{posisi nada D4}
(3,	116,	65,	14,	0,	0,	0),	{posisi nada D#4}
(3,	126,	75,	24,	0,	0,	0),	{posisi nada E4}
(3,	136,	85,	34,	0,	0,	0),	{posisi nada F4}
(3,	146,	95,	44,	0,	0,	0),	{posisi nada F#4}
(4,	156,	105,	54,	3,	0,	0),	{posisi nada G4}
(4,	166,	115,	64,	13,	0,	0),	{posisi nada G#4}
(4,	176,	125,	74,	23,	0,	0),	{posisi nada A4}
(4,	186,	135,	84,	33,	0,	0),	{posisi nada A#4}
(5,	196,	145,	94,	43,	2,	0),	{posisi nada B4}
(5,	206,	155,	104,	53,	12,	0),	{posisi nada C5}
(5,	216,	165,	114,	63,	22,	0),	{posisi nada C#5}
(5,	226,	175,	124,	73,	32,	0),	{posisi nada D5}
(5,	236,	185,	134,	83,	42,	0),	{posisi nada D#5}
(6,	246,	195,	144,	93,	52,	1),	{posisi nada E5}
(5,	205,	154,	103,	62,	11,	0),	{posisi nada F5}

(5,	215,	164,	113,	72,	21,	0),	{posisi nada F#5}
(5,	225,	174,	123,	82,	31,	0),	{posisi nada G5}
(5,	235,	184,	133,	92,	41,	0),	{posisi nada G#5}
(5,	245,	194,	143,	102,	51,	0),	{posisi nada A5}
(4,	204,	153,	112,	61,	0,	0),	{posisi nada A#5}
(4,	214,	163,	122,	71,	0,	0),	{posisi nada B5}
(4,	224,	173,	132,	81,	0,	0),	{posisi nada C6}
(4,	234,	183,	142,	91,	0,	0),	{posisi nada C#6}
(4,	244,	193,	152,	101,	0,	0),	{posisi nada D6}
(3,	203,	162,	111,	0,	0,	0),	{posisi nada D#6}
(3,	213,	172,	121,	0,	0,	0),	{posisi nada E6}
(3,	223,	182,	131,	0,	0,	0),	{posisi nada F6}
(3,	233,	192,	141,	0,	0,	0),	{posisi nada F#6}
(3,	243,	202,	151,	0,	0,	0),	{posisi nada G6}
(2,	212,	161,	0,	0,	0,	0),	{posisi nada G#6}
(2,	222,	171,	0,	0,	0,	0),	{posisi nada A6}
(2,	232,	181,	0,	0,	0,	0),	{posisi nada A#6}
(2,	242,	191,	0,	0,	0,	0),	{posisi nada B6}
(1,	201,	0,	0,	0,	0,	0),	{posisi nada C7}
(1,	211,	0,	0,	0,	0,	0),	{posisi nada C#7}
(1,	221,	0,	0,	0,	0,	0),	{posisi nada D7}
(1,	231,	0,	0,	0,	0,	0),	{posisi nada D#7}
(1,	241,	0,	0,	0,	0,	0))	{posisi nada E7}

Pada deklarasi konstanta tersebut, *ArrayNada* kolom 0 merupakan jumlah posisi nada pada fret gitar, sedangkan kolom 1 sampai 6 berisi data posisi. Format data posisi adalah $posisi\ fret \times 10 + posisi\ senar$. Misalnya nada G6, jumlah posisinya adalah 3, yaitu fret ke-24 senar no.3, fret ke-20 senar no.2, dan fret ke-15 senar no.1.

File temporary tersebut akan disimpan sebagai file GT2.TMP, dan secara otomatis file GT1.TMP akan dihapus karena sudah tidak dibutuhkan lagi dalam proses berikutnya.



4.2.4. Posisi Jari pada Fretboard

Data terakhir yang ditambahkan pada file hasil keluaran adalah posisi jari pada fretboard. Dimana yang digunakan adalah notasi jari tangan kiri, yaitu jari no.1 (telunjuk), no.2 (jari tengah), no.3 (jari manis), dan no.4 (kelingking). Data nomor jari tersebut ditambahkan pada file yang telah memuat data nada dan posisi. Selain itu juga ditambahkan header untuk pengenalan pada saat proses menampilkan data ke layar monitor. Struktur datanya adalah sebagai berikut :

<header><division><time signature><set tempo><event>+...<eof>

Header : 'Gtr'; {3 byte karakter ASCII}

Division : word; {pembagi delta time}

TimeSignature : D0 nn dd cc bb

D0 : nilai konstan;

nn : *nominator*, pembilang tanda birama

dd : *denominator*, 2^{dd} = penyebut tanda birama

cc : MIDI *clocks* untuk tiap ketukan metronome

bb : jumlah nada 1/32 tiap 24 MIDI clock (tiap nada 1/4)

SetTempo : C0 tt tt tt

C0 : nilai konstan

tt tt tt : 24 bit nilai tempo per nada 1/4 dalam mikrodetik

Event : <nothing | gitar note><delta-time>

Nothing : A0;

GitarNote : FF <length><note+position+finger>+

FF : nilai konstan;

Length : byte; {jumlah nada}

Note : byte; {data nada}

Position : byte; {data posisi}

Finger : byte; {data jari}

DeltaTime : (1-4 bytes); {variable-length quantity}

Eof : B0;

Format file di atas akan dikenali sebagai file .GTR. Adapun sebelum ada konfirmasi dari pengguna maka file tersebut masih dianggap sebagai file temporary dengan pengenalan GT3.TMP, yang formatnya sama dengan file .GTR tetapi tanpa header.

4.3. Modul Baca dan Konversi File MIDI

Modul ini digunakan untuk membuka file .MID yang merupakan masukan proses, mulai dari membaca header chunk, header track, dan data track, dilanjutkan mengkonversi data menjadi data yang lebih sederhana, yang hanya berisi data nada yang masuk dalam interval nada gitar 24 fret. Komponen data yang digunakan dalam modul ini adalah data notasi musik dalam format file MIDI dan data nada dalam interval nada gitar. Adapun struktur data yang dipakai seperti yang telah dijelaskan pada pembahasan perencanaan program. Berikut ini dijelaskan mengenai langkah-langkah pemrosesan yang digunakan dalam modul ini.

Proses diawali dengan prosedur untuk membuka file MIDI, yaitu memeriksa *header chunk* yang terdiri atas karakter ASCII 'MThd' (4 byte *chunk type*), panjang header chunk (4 byte *length*), format file (2 byte *format*), jumlah track (2 byte *ntrack*), dan pembagi delta-time (2 byte *division*). Jika file dalam format 2, atau pembagi delta-time menggunakan standar *SMPTE*, maka proses dibatalkan. Untuk file format 1, pembacaan track akan dibatasi sampai 25 track, selebihnya diabaikan.

Pada file format 0 yang hanya memiliki satu track, pembacaan track chunk dilakukan sesuai urutan yaitu header track yang diikuti oleh rangkaian event. Sedangkan pada file format 1, langkah selanjutnya adalah membaca header dari semua track yang akan dibaca, hal ini dilakukan untuk mengetahui track mana yang harus didahulukan. Pertama kali adalah membaca header dari track awal, yaitu dengan pengenalan karakter ASCII 'MTrk' (4 byte *chunk type*) dilanjutkan dengan panjang track chunk (4 byte *length*). Setelah itu membaca header track berikutnya, dimana posisi header untuk track berikutnya diketahui dari panjang track chunk yang sedang dibaca. Kemudian data event awal yang didapatkan dari masing-masing track disimpan dalam bentuk *array of record*. Adapun bentuk record tersebut dideklarasikan sebagai berikut :

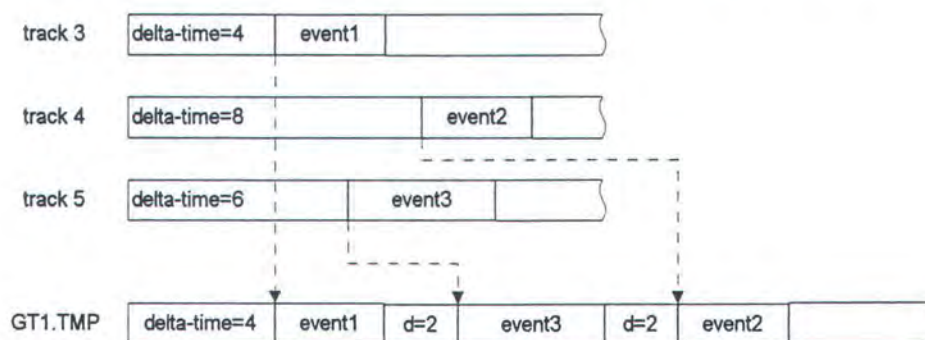
```

type
    RFormat1 = record
        Tdelay    : longint; {delta-time}
        TCurrPos  : longint; {posisi data yang dibaca}
        * TLastPos : longint; {posisi akhir track}
    end;

```

Langkah berikutnya adalah membaca event-event dari track chunk yang terdiri atas delta-time, tipe event, panjang data event, dan data event. Karena delta-time memiliki bentuk *variable-length quantity*, dengan panjang byte bervariasi antara 1-4 byte, maka untuk membaca delta-time memakai perulangan dengan kondisi : jika byte yang dibaca nilainya lebih besar atau sama dengan 128 (80 hex), maka byte berikutnya masih termasuk delta-time.

Pada file format 1, proses pembacaan event dilakukan secara vertikal, dalam arti pembacaan event pada semua track dilakukan secara ‘bersamaan’ (delta-time dari setiap event disimpan pada record yang telah disediakan). Proses urutan pembacaannya sesuai dengan nilai delta-time, dimana delta-time yang paling kecil yang didahulukan. Misalnya delta time satu event yang akan dibaca pada track 4 lebih besar dari delta-time event yang akan dibaca pada track 3, maka event pada track 3 yang akan dibaca lebih dahulu. Dengan demikian record *TDelay* untuk track 4 adalah delta-time track 4 dikurangi delta-time track 3. Sedangkan untuk record track 3 diisi delta-time beserta posisi event berikutnya. Pada gambar 4.1 ditunjukkan contoh penyalinan data file MIDI format 1. Dari contoh tersebut dapat diketahui bahwa proses pembacaan file format 1 mengikuti aturan antrian.



Gambar 4.1.

Penyalinan dan koversi data file MIDI format 1 ke dalam file penampung GT1.TMP

Setiap melakukan pembacaan event, data tersebut dikonversikan ke dalam struktur data nada dalam interval nada gitar yang disimpan dalam file penampung sementara GT1.TMP, dimana data-data yang diabaikan dan delta-time tidak sama dengan nol dinyatakan sebagai A0. Data-data yang digunakan untuk diproses lebih lanjut adalah *division* yang dinyatakan pada header chunk, delta-time, kemudian

event-event *set tempo* dengan tipe FF 51, *time signature* dengan tipe FF 58, dan *note on* (nada on) dengan tipe 90-9F.

4.4. Modul Pemilihan Posisi pada Fretboard

Modul ini digunakan untuk memproses lebih lanjut data yang ada pada file GT1.TMP. Data nada yang ada dalam GT1.TMP akan ditentukan data posisinya, kemudian data posisi tersebut ditambahkan dalam file sesuai struktur data yang telah dibahas pada perencanaan program. Langkah-langkah proses secara keseluruhan akan dijelaskan berikut ini.

Langkah awal dalam proses ini adalah membuka file GT1.TMP dan mempersiapkan file penampung GT2.TMP. Data awal berupa *division*, *time signature*, dan *set tempo* langsung disalin pada file penampung tanpa ada perubahan. Setelah data tersebut disalin (sebanyak 11 byte data), proses dilanjutkan dengan membaca event-event yang masing-masing diikuti delta-time. Delta-time masih dalam bentuk *variable-length quantity*, sehingga proses pembacaannya sama seperti pada proses pembacaan delta-time dalam file MIDI.

Event yang akan diproses lebih lanjut adalah yang mempunyai tipe FF (nada on dalam format GT1.TMP). Byte berikutnya dari FF adalah informasi jumlah nada yang akan dibunyikan dalam satu waktu. Data nada direlasikan dengan data posisi pada fretboard yang telah dideklarasikan sebagai nilai konstanta, seperti yang telah dijelaskan dalam pembahasan struktur data. Kemudian dilanjutkan dengan langkah-langkah sebagai berikut :

1. Inisialisasi nilai 0 untuk variabel *posisi prev* (posisi rata-rata interval sebelum interval yang akan diproses), dan inisialisasi nilai 0 untuk *jumlah nada tunggal*.
2. Proses pembacaan file GT1.TMP dilakukan per byte, kemudian dilanjutkan proses pembacaan sesuai dengan tipe data event yang sedang diproses. Jika event yang dibaca merupakan akhir file maka dilanjutkan langkah ke-9.
3. Jika data nada yang dibaca adalah nada tunggal maka dilanjutkan langkah ke-4, selain itu dilanjutkan langkah ke-8.
4. Jika *jumlah nada tunggal* bernilai 0 maka nilai *max interval* dan *min interval* sama dengan *data nada*, lanjutkan langkah ke-6, selain itu dilanjutkan langkah ke-5.
5. Jika nilai *max interval* kurang dari nilai *data nada* maka nilai *max interval* sama dengan nilai *data nada*. Selain itu jika nilai *min interval* lebih dari nilai *data nada* maka nilai *min interval* sama dengan nilai *data nada*. Lanjutkan langkah ke-6.
6. Nilai *jumlah nada tunggal* ditambah satu. Jika selisih *max interval* dan *min interval* lebih dari 12 (jumlah nada satu oktaf), maka dilanjutkan langkah ke-7, selain itu dilanjutkan langkah ke-2.
7. Nilai *jumlah nada tunggal* dikurangi satu, kemudian dilanjutkan dengan **proses pencarian jarak terpendek** (akan dibahas lebih lanjut). Nilai *jumlah nada tunggal* diset 1. Kembali ke langkah ke-2.



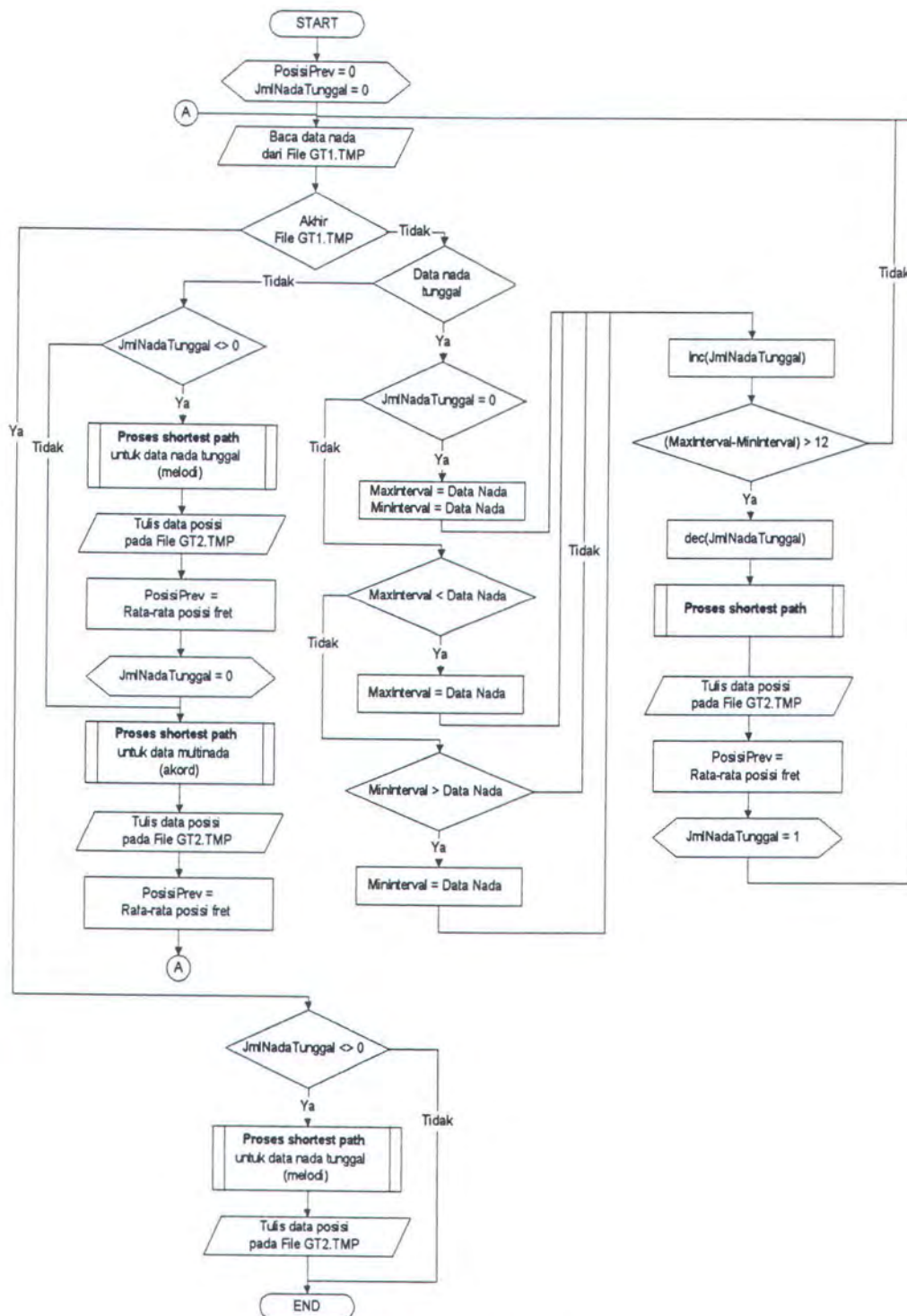
8. Jika *jumlah nada tunggal* tidak sama dengan 0 maka dilanjutkan **proses pencarian jarak terpendek** untuk data-data nada tunggal sebelumnya (melodi) kemudian *jumlah nada tunggal* diset 0, lalu diikuti dengan **proses pencarian jarak terpendek** untuk data multinada (akord) yang sedang diproses. Jika *jumlah nada tunggal* sama dengan 0 maka langsung menuju **proses pencarian jarak terpendek** untuk data multinada. Kembali ke langkah ke-2.
9. Jika *jumlah nada tunggal* tidak sama dengan 0 maka dilanjutkan **proses pencarian jarak terpendek** untuk data-data nada tunggal sebelumnya (melodi).

Setiap melakukan proses pencarian jarak terpendek maka selalu diakhiri dengan menyimpan data posisi nada ke dalam file GT2.TMP, dan *posisi prev* berubah menjadi sama dengan rata-rata posisi fret pada interval nada yang baru diproses.

Pada gambar 4.2 ditunjukkan diagram alir dari proses pemilihan posisi pada fret board seperti yang telah dibahas di atas. Kemudian pada gambar 4.3 ditunjukkan diagram alir dari proses pencarian jarak terpendek. Pembahasan lebih lanjut mengenai langkah-langkah proses pencarian jarak terpendek akan dijelaskan berikut ini.

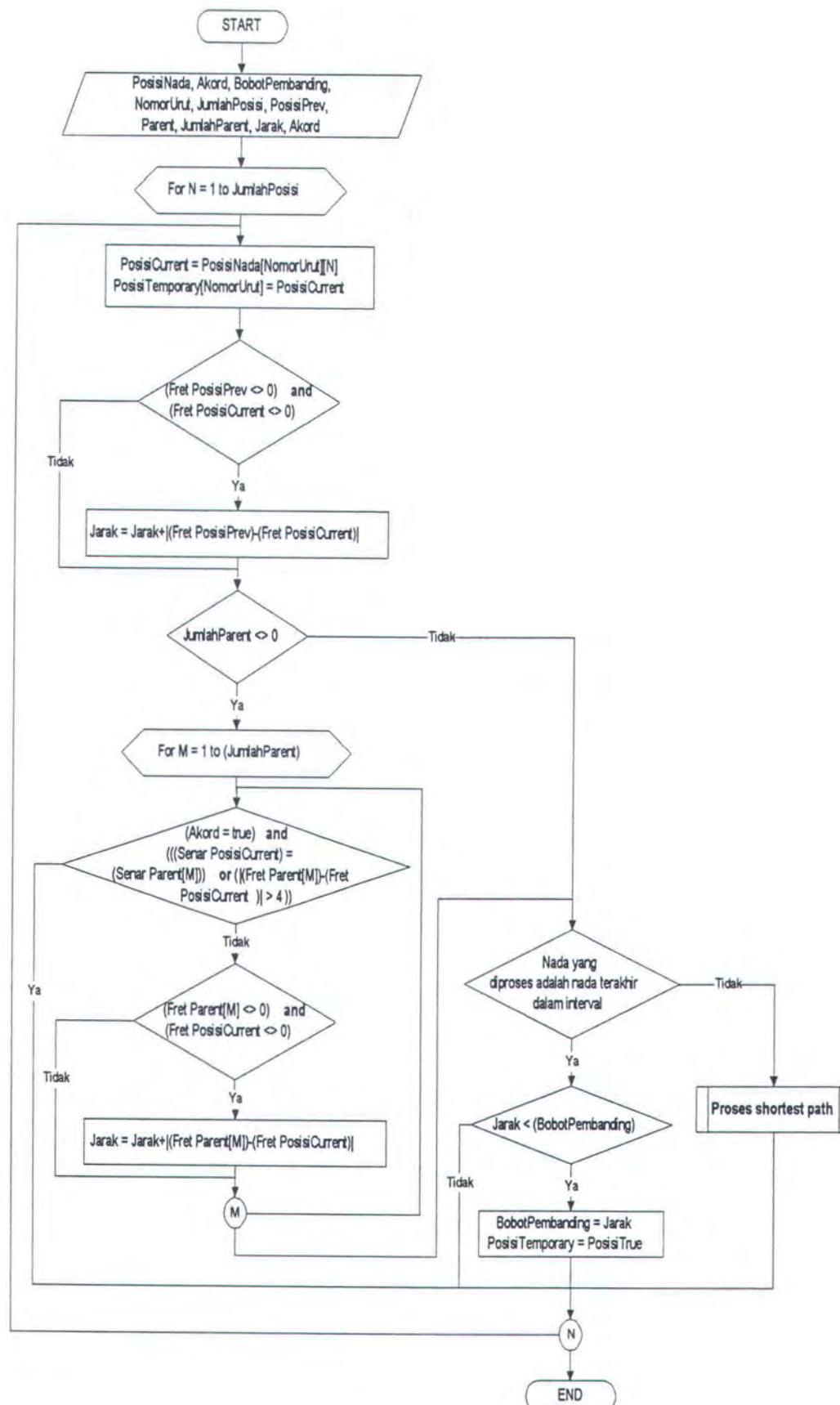
Sebagaimana telah dijelaskan pada Bab II, bahwa batasan interval yang digunakan pada program tugas akhir ini adalah interval satu oktaf. Maka pada pembahasan pencarian jarak terpendek, yang dimaksud dengan interval adalah

batasan interval yang telah ditentukan tersebut, bukan interval untuk keseluruhan melodi (interval dalam suatu melodi bisa lebih dari satu oktaf).



Gambar 4.2.

Diagram alir proses pemilihan posisi pada fretboard



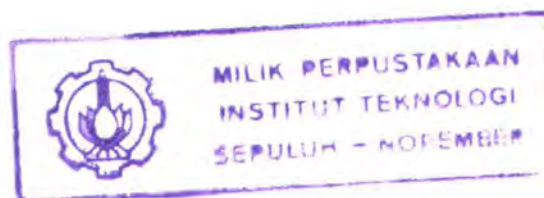
Gambar 4.3.

Diagram alir proses pencarian jarak terpendek (*Shortest Path*)

Proses pencarian jarak terpendek :

Proses ini membutuhkan data-data yang berupa *parameter formal* maupun *parameter nilai*. Data-data yang termasuk parameter formal adalah *posisi nada* yang memiliki tipe data array dua dimensi, *akord* yang berupa data boolean untuk membedakan apakah data yang diproses berupa melodi atau akord, dan *bobot pembeding*. Sedangkan data-data yang termasuk parameter nilai adalah *nomor urut nada* dalam satu interval, *jumlah posisi* untuk nada tersebut, *posisi prev* yang merupakan posisi rata-rata interval sebelumnya, posisi-posisi *parent* yaitu posisi nada sebelumnya dalam satu interval, *jumlah parent*, serta *jarak* yang merupakan hasil penjumlahan jarak antara posisi-posisi *parent*. Adapun langkah-langkah proses ini adalah sebagai berikut :

1. Untuk $N = 1$ sampai dengan *jumlah posisi* maka lakukan langkah ke-2.
2. *Posisi current* (posisi yang akan diproses) adalah sama dengan *posisi nada* yang ke-(*nomor urut*, N). *Posisi temporary[nomor urut]* sama dengan *posisi current*. Jika *posisi prev* tidak pada fret ke-0 dan *posisi current* tidak pada fret ke-0, maka *jarak* adalah *jarak* sebelumnya ditambah jarak antara *posisi prev* dengan *posisi current*. Jika *jumlah parent* tidak sama dengan 0 (bukan nada awal dalam satu interval yang sedang diproses) maka dilanjutkan langkah ke-3, jika kondisi tersebut tidak terpenuhi maka dilanjutkan langkah ke-6.
3. Untuk $M = 1$ sampai dengan *jumlah parent* lakukan langkah ke-4. Selanjutnya lakukan langkah ke-6.



4. Jika yang dicari berupa data multinada / akord (nada-nada yang dibunyikan bersamaan), dan senar pada *posisi current* sama dengan senar pada *parent* ke-M atau jarak antara *posisi current* dengan posisi *parent* ke-M lebih dari 4 fret, maka proses tidak dilanjutkan, kembali pada langkah ke-2. Jika kondisi tersebut tidak terpenuhi maka dilanjutkan langkah ke-5.
5. Jika posisi *parent* ke-M tidak pada fret ke-0 dan *posisi current* tidak pada fret ke-0, maka *jarak* adalah *jarak* sebelumnya ditambah jarak antara *posisi current* dengan posisi *parent* ke-M.
6. Jika nada yang diproses bukan nada terakhir dalam satu interval, maka dilanjutkan **proses pencarian jarak terpendek** (proses rekursif) untuk nada berikutnya dimana *posisi current* termasuk posisi *parent*. Jika kondisi ini tidak terpenuhi, dilanjutkan dengan langkah ke-7.
7. Jika nilai *jarak* lebih kecil dari nilai *bobot pembanding*, maka nilai *jarak* akan dipakai sebagai *bobot pembanding* untuk proses selanjutnya.

Dari langkah-langkah yang telah dibahas tersebut akan didapatkan posisi-posisi untuk semua data nada dalam file GT1.TMP, dimana untuk selanjutnya disimpan dalam file GT2.TMP sebagai data posisi nada pada fretboard dengan struktur data yang telah dibahas sebelumnya.

4.5. Modul Penentuan Jari untuk Tiap Posisi

Proses konversi data yang terakhir dilakukan pada modul ini. Proses tersebut berhubungan dengan penentuan jari untuk posisi-posisi yang telah

didapatkan melalui proses pencarian jarak terdekat, yaitu data yang telah disimpan dalam file GT2.TMP.

Dalam penentuan jari ini, proses dibedakan menjadi 2, yaitu proses penentuan jari untuk akord, dan proses penentuan jari untuk melodi. Berikut ini adalah penjelasan langkah-langkah pemrosesan data tersebut.

Langkah-langkah penentuan jari untuk akord :

1. Pada langkah awal dipersiapkan empat variabel penampung bertipe *array*, yaitu *temp1*, *temp2*, dan *temp3*. *Temp1* digunakan untuk penampung data posisi sesuai urutan data dalam file GT2.TMP.
2. Data yang terdapat pada *temp1* disalin ke dalam *temp2*, untuk kemudian data dalam *temp2* tersebut diurutkan sesuai dengan urutan fret, mulai dari fret yang terdepan (nomor kecil) sampai fret yang terakhir (nomor besar) dalam satu interval yang akan dicari.
3. Setelah itu dilanjutkan pengurutan data dalam *temp2* sesuai urutan senar dalam satu fret, mulai dari senar yang terbesar (teratas) sampai dengan senar yang terkecil (terbawah). Hal ini berkaitan dengan penempatan posisi jari yang menganut suatu aturan prioritas sesuai dengan urutan fret dan senar tersebut, yaitu dimulai jari no.1 (telunjuk), no.2 (jari tengah), no.3 (jari manis), sampai yang terakhir no.4 (kelingking).
4. *Temp3* digunakan untuk penampung data jari yang digunakan sesuai urutan data posisi pada *temp2*. Inisialisasi nilai 0 untuk variabel *jari satu* yang digunakan untuk mengetahui pada fret seberapa jari kesatu digunakan.

Variabel *tanda* yang berupa data boolean diset *false* digunakan untuk mengetahui ada tidaknya posisi pada fret ke-0.

5. Jika data pertama pada *temp2* memiliki posisi fret ke-0, maka dilanjutkan langkah ke-6, selain itu dilanjutkan langkah ke-8
6. Untuk $N = 1$ sampai dengan *jumlah nada* lakukan langkah ke-7. Selanjutnya lakukan langkah ke-8.
7. Jika data ke-N pada *temp2* adalah data dengan posisi fret ke-0, maka tidak ada jari yang digunakan pada posisi tersebut, nilai *temp3* ke N sama dengan 0 dan *tanda* bernilai *true*. Jika data ke-N pada *temp2* bukan pada posisi fret ke-0 dan *tanda* bernilai *true*, maka *tanda* bernilai *false* dan *jari satu* sama dengan N.
8. Jika *tanda* bernilai *true* maka dilanjutkan langkah ke-22.
9. Variabel *atas* berupa data boolean diset *false* digunakan untuk mengetahui penggunaan jari kesatu pada posisi senar teratas dalam satu fret. Jika *jari satu* sama dengan 0 maka *X* sama dengan 1, jika tidak maka *X* sama dengan *jari satu*. Jika *jari satu* tidak sama dengan 0, dan data senar pada *temp2[X]* lebih besar daripada data senar *temp2[jari satu - 1]*, maka *atas* bernilai *true*.
10. Untuk $N = X$ sampai dengan *jumlah nada* lakukan langkah ke-11, selanjutnya lakukan langkah ke-22.
11. *Jarak* adalah jarak antara fret *temp2[N]* dengan fret *temp2[N-1]*, dilanjutkan langkah ke-12.
12. Nilai data ke-N *temp3* sama dengan 1, jika kondisi-kondisi berikut ini terpenuhi (jika tidak terpenuhi dilanjutkan langkah ke-13) :

- a. Posisi fret data ke- N pada *temp2* sama dengan posisi fret data ke- X pada *temp2*.
 - b. Nilai *jari satu* sama dengan 0, atau nilai *jari satu* tidak sama dengan 0 tetapi jika salah satu kondisi berikut ini terpenuhi :
 - N sama dengan X .
 - Data senar pada *temp2*[N] lebih kecil dari pada data senar pada *temp2*[*jari satu* - 1], dan *atas* bernilai *false*.
13. Jika *temp3*[$N-1$] sama dengan 1, maka dilanjutkan langkah ke-14, jika tidak maka dilanjutkan langkah ke-18.
 14. Jika jumlah nada berikutnya sama dengan 0 maka dilanjutkan langkah ke-15, jika tidak maka dilanjutkan langkah ke-16.
 15. Jika jarak kurang dari sama dengan 1 maka *temp3*[N] sama dengan 2, selain itu jika jarak sama dengan 2 maka *temp3*[N] sama dengan 3, selain itu *temp3*[N] sama dengan 4.
 16. Jika jumlah nada berikutnya sama dengan 1 maka dilanjutkan langkah ke-17, jika tidak maka *temp3*[N] sama dengan 2.
 17. Jika jarak kurang dari sama dengan 1 maka *temp3*[N] sama dengan 2, selain itu *temp3*[N] sama dengan 3.
 18. Jika *temp3*[$N-1$] sama dengan 2 maka dilanjutkan langkah ke-19, jika tidak maka dilanjutkan langkah ke-21.
 19. Jika jumlah nada berikutnya sama dengan 0 maka dilanjutkan langkah 20, selain itu *temp3*[N] sama dengan 4.

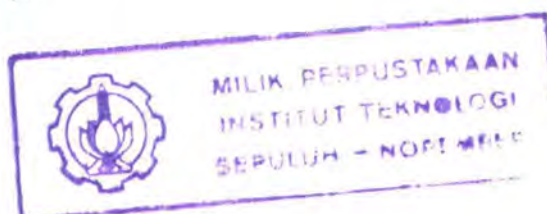
20. Jika jarak kurang dari sama dengan 1 maka $temp3[N]$ sama dengan 3.
21. Jika $temp3[N-1]$ sama dengan 3 dan jarak kurang dari sama dengan 1 maka $temp3[N]$ sama dengan 4, jika tidak maka $temp3[N]$ sama dengan 6 (digunakan untuk menandai posisi yang akan diabaikan).
22. Mengurutkan data nomor jari yang ada pada $temp3$ yang disesuaikan dengan urutan posisi yang ada pada $temp1$.

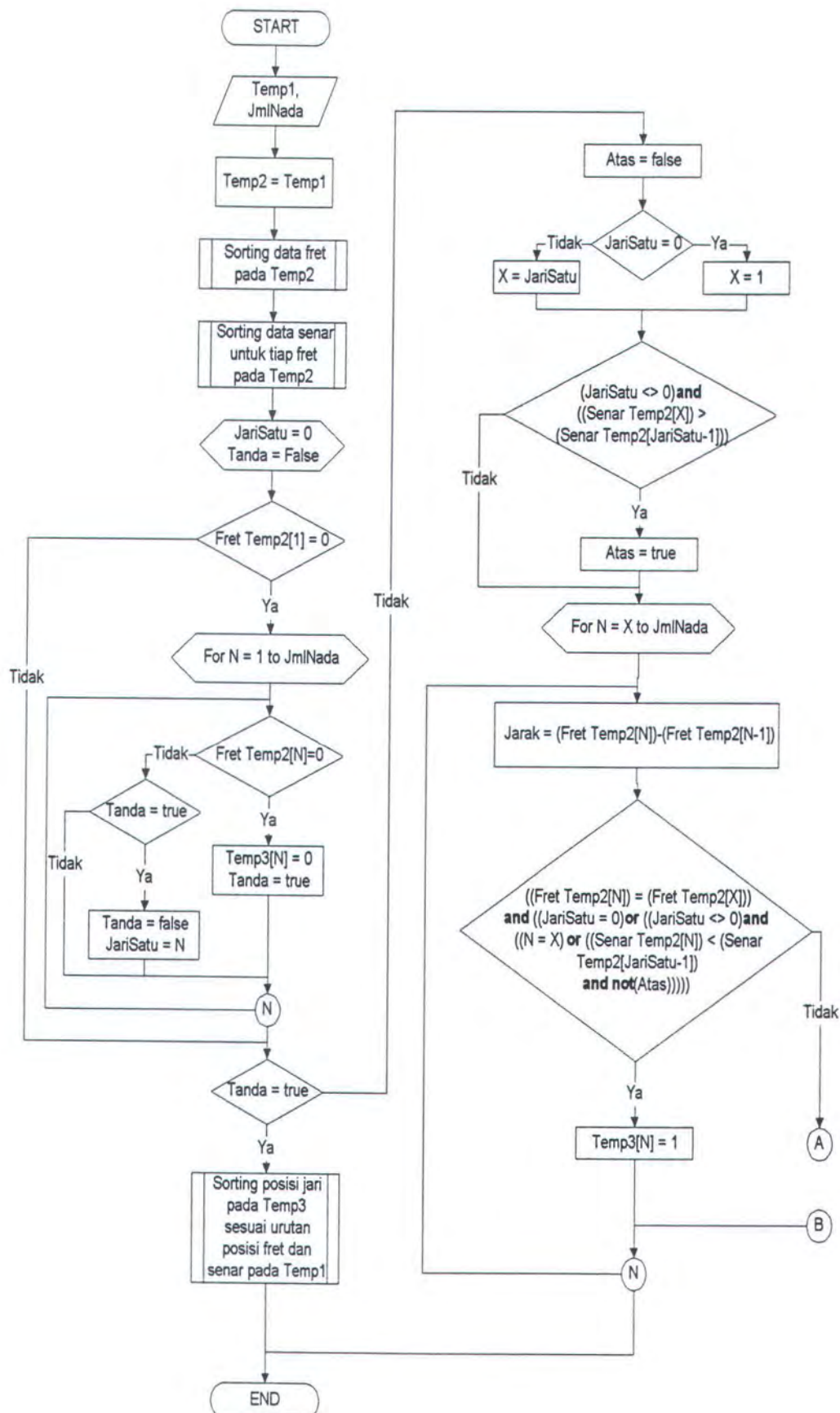
Langkah-langkah penentuan jari untuk melodi :

Proses penentuan jari untuk melodi lebih sederhana daripada proses penentuan jari untuk akord.

Untuk penentuan jari pada posisi awal melodi, dicari berdasarkan posisi nada berikutnya (posisi nada kedua). Tetapi jika posisi awal melodi tersebut pada fret ke-0 maka tidak ada jari yang digunakan pada posisi tersebut, sehingga posisi yang dianggap sebagai posisi awal adalah posisi berikutnya. Jika jari untuk posisi awal telah diketahui maka proses penentuan jari berikutnya dilakukan berdasarkan data jari maupun posisi nada sebelumnya.

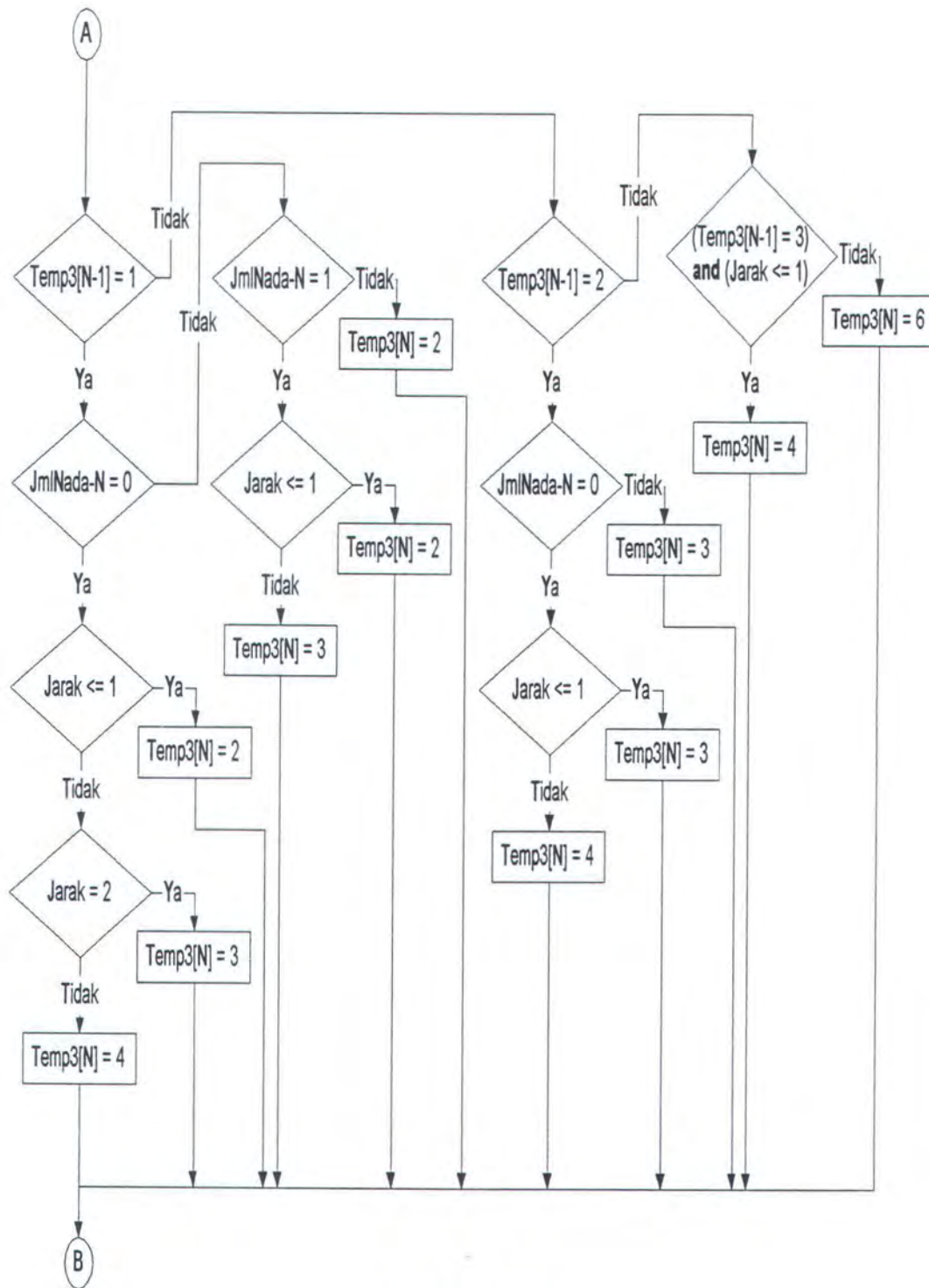
Jika data sebelumnya adalah data multinada (akord), maka dipilih posisi jari dari akord tersebut yang terdekat dengan posisi nada tunggal pada fretboard yang sedang diproses. Jika data sebelumnya adalah data nada tunggal dan posisi fretnya tidak pada fret ke-0, maka data yang digunakan adalah data posisi jari sebelumnya, dan nilai hasil pengurangan antara posisi fret yang sedang diproses dengan posisi fret sebelumnya (bisa bernilai negatif).





Gambar 4.4a.

Diagram alir proses penentuan posisi jari untuk akord



Gambar 4.4b.

Diagram alir proses penentuan posisi jari untuk akord (lanjutan)

Berikut ini adalah kondisi-kondisi yang ada jika posisi fret sekarang maupun posisi fret sebelumnya tidak pada posisi fret ke-0, dan data sebelumnya adalah data nada tunggal :

Jari Sebelum	Jarak	Jari Sekarang
1	≤ 0	1
1	$= 1$	2
1	$= 2$	3
1	≥ 3	4
2	< 0	1
2	$= 0$	2
2	$= 1$	3
2	≥ 2	4
3	< -1	1
3	$= -1$	2
3	$= 0$	3
3	≥ 1	4
4	< -2	1
4	$= -2$	2
4	$= -1$	3
4	$= 0$	4

Data-data yang telah diproses tersebut disimpan sementara dalam file GT3.TMP, jika akan disimpan untuk seterusnya maka akan diubah menjadi file *.GTR dengan penambahan header, sesuai dengan struktur data yang telah dibahas sebelumnya.

4.6. Modul Pengolah Tampilan File Hasil Keluaran

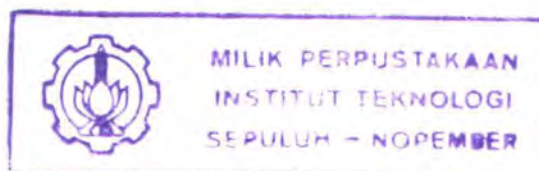
Modul ini menangani proses menampilkan data-data pada file GT3.TMP atau file *.GTR. Proses tersebut terdiri atas langkah-langkah berikut :

1. Membuka file GT3.TMP atau *.GTR, kemudian membaca event-event yang ada. Delay yang digunakan untuk waktu menampilkan suatu posisi, dihitung dengan mengalikan data *delta-time* dengan waktu tempo (dari data *set tempo*)

kemudian dibagi dengan *division*. Waktu delay tersebut dinyatakan dalam satuan mikrodetik.

2. Data nada digunakan untuk menampilkan notasi pada titi nada, dalam hal ini masih berupa tampilan sederhana, yaitu ditampilkan sebagai nada penuh. Tampilan ini berfungsi untuk menunjukkan pada titi nada yang ke berapa notasi nada yang direpresentasikan sebagai posisi pada fretboard.
3. Data posisi dan nomor jari ditampilkan sebagai notasi jari pada fretboard. Waktu untuk menampilkan setiap posisi secara *default* disesuaikan dengan delta-time pada setiap event.

Proses menampilkan data hasil keluaran ini dirancang dengan menentukan posisi-posisi yang telah ditentukan pada layar monitor, berdasarkan satuan nilai konstanta yang akan dikalikan dengan bagian dari data yang ada, sehingga data tersebut dapat ditampilkan pada posisi yang tepat. Hal ini berhubungan dengan perhitungan jumlah piksel yang ada pada layar monitor.



BAB V

ANALISA PROGRAM

Dalam bab ini akan dibahas mengenai jalan program, hasil keluaran atau *output*, serta analisa dan evaluasi dari program tugas akhir ini.

5.1. Jalan dan Keluaran Program

Proses utama dalam program yang dibuat adalah proses konversi data, yaitu dari data notasi musik dalam format file MIDI menjadi data posisi jari pada fretboard. Adapun dalam program ini terdapat tiga proses konversi data, seperti yang telah diuraikan pada pembahasan sebelumnya.

Data-data hasil konversi disimpan ke dalam file-file penampung yaitu GT1.TMP yang berisi data nada yang termasuk dalam interval nada gitar yang merupakan penyederhanaan dari file MIDI yang diproses, GT2.TMP yang merupakan salinan GT1.TMP yang dilengkapi data posisi pada fretboard untuk tiap nada, dan GT3.TMP yang merupakan salinan GT2.TMP yang dilengkapi data jari untuk tiap posisi nada.

Hasil keluaran akhir dari program ini berupa tampilan posisi jari pada fret gitar pada layar monitor. Adapun tampilan tersebut berdasarkan data yang merupakan hasil proses konversi file notasi musik.

Berikut ini adalah contoh dari notasi musik yang akan diproses serta format penyimpanannya dalam bentuk file MIDI. Kemudian contoh dari file GT1.TMP,

GT2.TMP, dan GT3.TMP, serta contoh sebagian dari bentuk tampilan yang dihasilkan. Adapun perbaikan dan penyempurnaan program ini berdasarkan pada perbandingan dari data-data yang ada pada setiap file.



Gambar 5.1.

Notasi musik yang akan diterjemahkan

Gambar 5.1 menunjukkan notasi musik yang akan diterjemahkan. Representasi notasi musik tersebut dalam bentuk file MIDI format 1 adalah sebagai berikut :

Header chunk :

4D 54 68 64	Mthd
00 00 00 06	panjang header chunk
00 01	format 1
00 02	2 track
00 F0	240 ketukan tiap nada $\frac{1}{4}$

Pertama track chunk untuk track time signature / tempo :

4D 54 72 6B	Mtrk
00 00 00 13	panjang track chunk (19)

Delta-Time	Event	Keterangan
00	FF 58 04 06 03 24 08	time signature
00	FF 51 03 09 27 C0	tempo
00	FF 2F 00	akhir track

Kemudian track chunk untuk track musik :

4D 54 72 6B	Mtrk
00 00 00 B8	panjang track chunk (184)

Delta-Time	Event	Keterangan
84 58	90 3E 50	nada on D ₅
78	80 3E 40	nada off D ₅
00	90 3E 50	nada on D ₅
00	90 3A 50	nada on A _{#4}
00	90 2B 50	nada on G ₃
00	90 32 50	nada on D ₄
60	80 3A 40	nada off A _{#4}
00	80 3E 40	nada off D ₅
18	90 43 50	nada on G ₅
48	80 32 40	nada off D ₄
00	80 2B 40	nada off G ₃
18	80 43 40	nada off G ₅
18	90 43 50	nada on G ₅
60	80 43 40	nada off G ₅
18	90 32 50	nada on D ₄
00	90 2B 50	nada on G ₃
00	90 3A 50	nada on A _{#4}
00	90 43 50	nada on G ₅
81 40	80 43 40	nada off G ₅
00	80 3A 40	nada off A _{#4}
00	80 2B 40	nada off G ₃
00	80 32 40	nada off D ₄
30	90 45 50	nada on A ₅
60	80 45 40	nada off A ₅
18	90 2B 50	nada on G ₃
00	90 32 50	nada on D ₄
00	90 3E 50	nada on D ₅
00	90 46 50	nada on A _{#5}
81 40	80 46 40	nada off A _{#5}
00	80 3E 40	nada off D ₅
00	80 32 40	nada off D ₄
00	80 2B 40	nada off G ₃
30	90 45 50	nada on A ₅
60	80 45 40	nada off A ₅
18	90 32 50	nada on D ₄
00	90 2B 50	nada on G ₃
00	90 3E 50	nada on D ₅
00	90 46 50	nada on A _{#5}
81 40	80 46 40	nada off A _{#5}
00	80 3E 40	nada off D ₅

00	80 2B 40	nada off G ₃
00	80 32 40	nada off D ₄
30	90 43 50	nada on G ₅
60	80 43 40	nada off G ₅
00	FF 2F 00	akhir track

File MIDI tersebut dibaca, kemudian dikonversi menjadi file GT1.TMP :

00 F0		division
D0 06 03 24 08		time signature
C0 09 27 C0		set tempo
Event	Delta-Time	Keterangan
	84 58	
FF 01 3E	78	1 nada : D ₅
FF 04 3E 3A 2B 32	60	4 nada : D ₅ , A _{#4} , G ₃ , D ₄
A0	18	<i>nothing</i>
FF 01 43	48	1 nada : G ₅
A0	18	<i>nothing</i>
A0	18	<i>nothing</i>
FF 01 43	60	1 nada : G ₅
A0	18	<i>nothing</i>
FF 04 32 2B 3A 43	81 40	4 nada : D ₄ , G ₃ , A _{#4} , G ₅
A0	30	<i>nothing</i>
FF 01 45	60	1 nada : A ₅
A0	18	<i>nothing</i>
FF 04 2B 32 3E 46	81 40	4 nada : G ₃ , D ₄ , D ₅ , A _{#5}
A0	30	<i>nothing</i>
FF 01 45	60	1 nada : A ₅
A0	18	<i>nothing</i>
FF 04 32 2B 3E 46	81 40	4 nada : D ₄ , G ₃ , D ₅ , A _{#5}
A0	30	<i>nothing</i>
FF 01 43	60	1 nada : G ₅
B0		akhir file

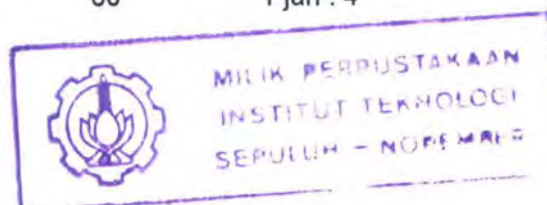
Yang kedua adalah file GT2.TMP, yang merupakan hasil pemrosesan file GT1.TMP pada contoh yang pertama, data yang ditambahkan adalah data posisi :

00 F0	division
D0 06 03 24 08	time signature
C0 09 27 C0	set tempo

Event	Delta-Time	Keterangan
	84 58	
FF 01 3E 32	78	1 posisi : 3.2
FF 04 3E 32 3A 33 2B 36 32 04	60	4 posisi : 3.2, 3.3, 3.6, 0.4
A0	18	<i>nothing</i>
FF 01 43 31	48	1 posisi : 3.1
A0	18	<i>nothing</i>
A0	18	<i>nothing</i>
FF 01 43 31	60	1 posisi : 3.1
A0	18	<i>nothing</i>
FF 04 32 04 2B 36 3A 33 43 31	81 40	4 posisi : 0.4, 3.6, 3.3, 3.1
A0	30	<i>nothing</i>
FF 01 45 51	60	1 posisi : 5.1
A0	18	<i>nothing</i>
FF 04 2B 36 32 55 3E 32 46 61	81 40	4 posisi : 3.6, 5.5, 3.2, 6.1
A0	30	<i>nothing</i>
FF 01 45 51	60	1 posisi : 5.1
A0	18	<i>nothing</i>
FF 04 32 55 2B 36 3E 32 46 61	81 40	4 posisi : 5.5, 3.6, 3.2, 6.1
A0	30	<i>nothing</i>
FF 01 43 31	60	1 posisi : 3.1
B0		akhir file

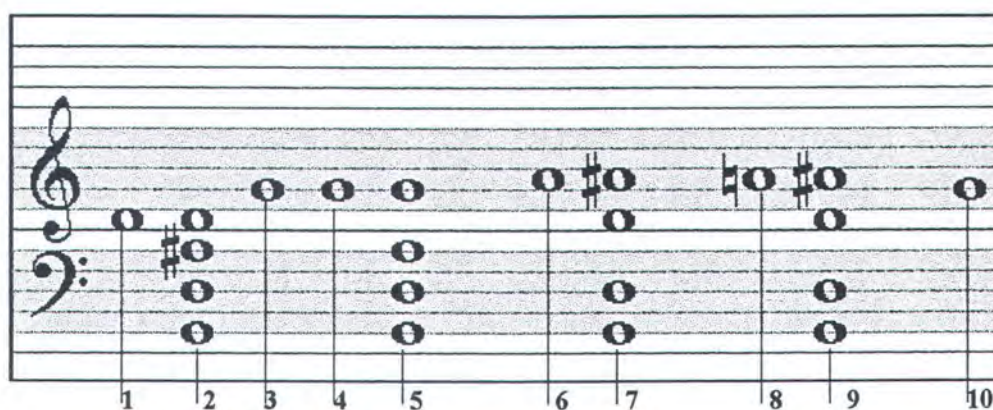
File penampung yang terakhir adalah GT3.TMP, contoh berikut adalah hasil pemrosesan lebih lanjut dari file GT2.TMP pada contoh di atas, data yang ditambahkan adalah data nomor jari :

00 F0		division
D0 06 03 24 08		time signature
C0 09 27 C0		set tempo
Event	Delta-Time	Keterangan
	84 58	
FF 01 3E 32 03	78	
FF 04 3E 32 03 3A 33 02 2B 36 01 32 04 00	60	3 jari : 3, 2, 1, 0
A0	18	<i>nothing</i>
FF 01 43 31 04	48	1 jari : 4
A0	18	<i>nothing</i>
A0	18	<i>nothing</i>
FF 01 43 31 04	60	1 jari : 4



A0	18	<i>nothing</i>
FF 04 32 04 00 2B 36 01 3A 33 02 43 31 03	81 40	3 jari : 0, 1, 2, 3
A0	30	<i>nothing</i>
FF 01 45 51 04	60	1 jari : 4
A0	18	<i>nothing</i>
FF 04 2B 36 01 32 55 03 3E 32 01 46 61 04	81 40	3 jari : 1, 3, 1, 4
A0	30	<i>nothing</i>
FF 0145 51 04	60	1 jari : 4
A0	18	<i>nothing</i>
FF 04 32 55 03 2B 36 01 3E 32 01 46 61 04	81 40	3 jari : 3, 1, 1, 4
A0	30	<i>nothing</i>
FF 01 43 31 01	60	1 jari : 1
B0		akhir file

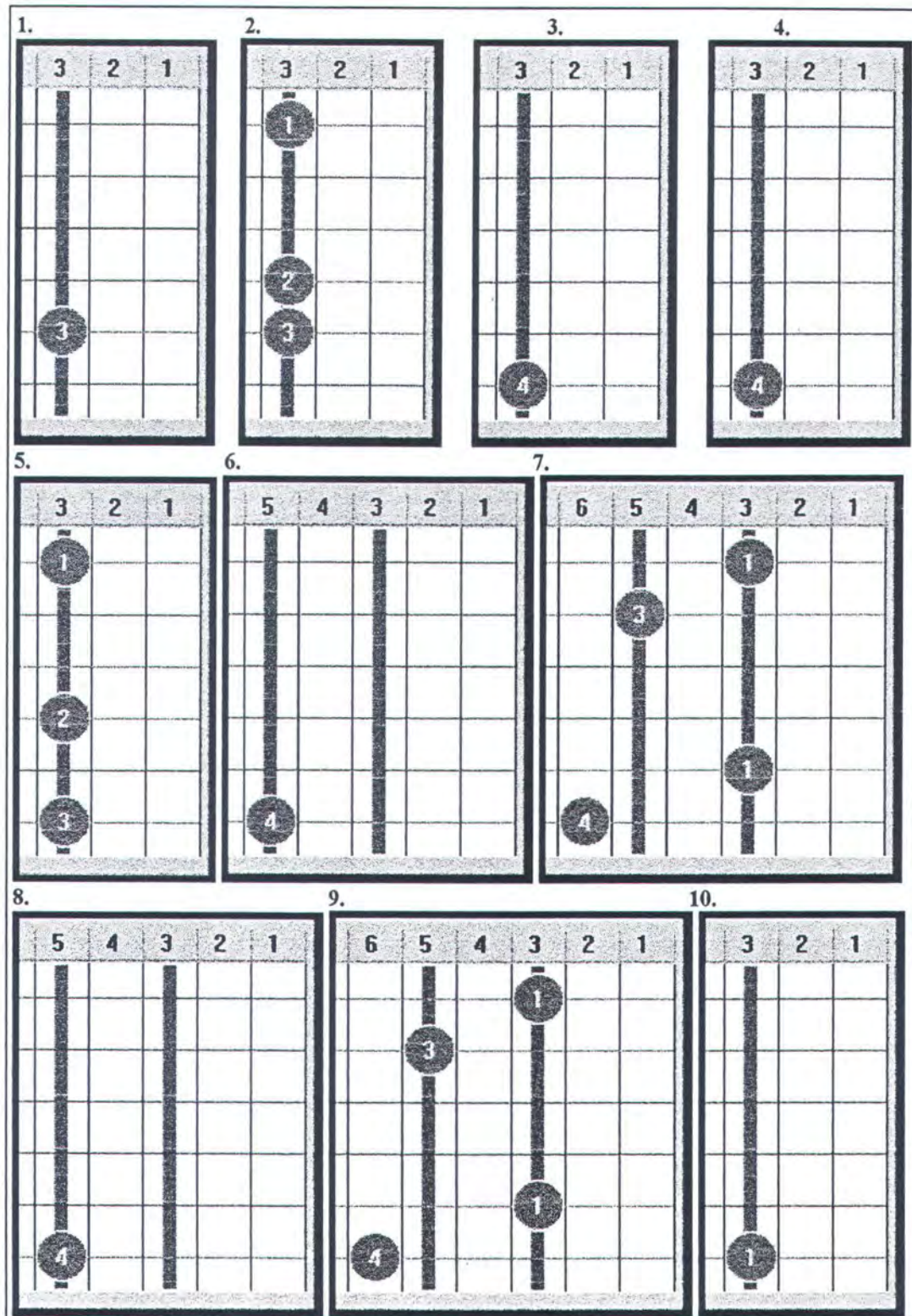
Dari file GT3.TMP, jika ingin disimpan dalam bentuk *.GTR akan ditambahkan header file, yaitu berupa tambahan karakter ASCII 'Gtr'.



Gambar 5.2.

Bentuk tampilan notasi musik (nada) yang berhasil diterjemahkan

Gambar 5.2 menunjukkan nada-nada yang berhasil diterjemahkan, kebetulan nada-nada pada notasi musik yang terdapat pada gambar 5.1 dapat diterjemahkan secara keseluruhan tanpa ada nada yang diabaikan. Angka 1 sampai dengan 10 digunakan untuk menandai urutan posisi-posisi seperti yang ditunjukkan pada gambar 5.3, yang merupakan sebagian dari bentuk tampilan posisi jari pada fretboard yang dihasilkan pada layar monitor.



Gambar 5.3.

Bentuk tampilan posisi jari tangan pada fretboard

Untuk kasus tertentu ada kemungkinan terdapat perubahan / pengabaian data nada baik dari GT1.TMP ke GT2.TMP, atau dari GT2.TMP ke GT3.TMP.

Pada proses penentuan posisi nada pada fretboard, data nada yang diabaikan adalah nada-nada yang tidak mungkin dibunyikan bersamaan karena memiliki posisi senar yang sama atau jarak posisi fret yang terlalu jauh (lebih besar dari 5 fret). Sedangkan data nada yang diabaikan pada proses penentuan jari untuk tiap posisi pada fretboard adalah nada-nada yang tidak mungkin dibunyikan bersamaan karena membutuhkan lebih dari empat jari untuk merepresentasikannya.

5.2. Analisa Dan Evaluasi

Pada dasarnya program ini dirancang hanya untuk notasi musik yang sesuai dan bisa direpresentasikan pada permainan gitar, yaitu dengan batasan interval nada antara E_3 sampai dengan E_7 . Namun mengingat banyak file MIDI yang berisi notasi musik yang sebenarnya digunakan untuk alat musik lain atau orkestra yang memiliki batasan nada di luar interval nada gitar, maka program ini dikondisikan untuk bisa memproses notasi musik tersebut dengan konsekuensi hasil penerjemahan akan terdapat data nada yang diabaikan / tidak diproses. Tentu hal tersebut akan menimbulkan penerjemahan yang tidak lengkap / tidak sesuai dengan notasi musik masukan.

Tingkat perubahan data nada dari file masukan menjadi file hasil penerjemahan tergantung dari bentuk notasi musik yang diproses. Perubahan data tersebut bisa disebabkan oleh :

- Jumlah dan macam event yang digunakan
- Batasan interval nada pada notasi musik



- Jumlah *chanel* atau alat musik yang digunakan
- Jumlah dan kondisi nada-nada yang dibunyikan pada saat yang sama

Kondisi-kondisi diatas akan disesuaikan dengan kemampuan dan kondisi alat musik gitar maupun pemainnya. Jadi jika kondisi tersebut sesuai maka notasi musik yang diproses akan bisa diterjemahkan secara keseluruhan dan tidak mengalami perubahan. Dengan demikian besarnya data hasil keluaran tidak tergantung pada besarnya data notasi musik masukan, tetapi tergantung pada data nada notasi musik yang memenuhi kondisi yang sesuai dengan alat musik gitar. Hubungan kedua elemen tersebut memiliki fungsi linear, dimana untuk setiap data nada yang diterjemahkan akan bertambah 2 byte yaitu satu byte untuk data posisi pada fretboard dan satu byte berikutnya untuk data nomor jari yang digunakan.

Untuk mengetahui jumlah nada yang diabaikan, maka pada setiap proses konversi disertakan *counter* untuk menghitung jumlah nada yang diabaikan. Pada konversi file MIDI ke GT1.TMP dibutuhkan dua variabel sebagai *counter* untuk menghitung *jumlah nada yang ada* serta *jumlah nada yang diluar interval nada gitar*. Pada konversi file GT1.TMP ke GT2.TMP dibutuhkan satu variabel sebagai *counter* untuk menghitung *jumlah nada yang tidak mungkin dibunyikan bersamaan* karena memiliki *posisi senar yang sama* atau *jarak posisi fret yang terlalu jauh* (lebih besar dari 5 fret). Kemudian pada konversi file GT2.TMP ke GT3.TMP dibutuhkan satu variabel sebagai *counter* untuk menghitung *jumlah nada yang tidak mungkin dibunyikan bersamaan* karena *membutuhkan lebih dari empat jari* untuk merepresentasikannya.

Kesalahan yang mungkin timbul pada saat membaca file MIDI, mungkin disebabkan file tersebut mengandung data *real-time message* yang bisa disisipkan pada event yang lain. Biasanya data tersebut digunakan untuk pengolah suara drum atau perkusi, yang sebenarnya tidak digunakan pada proses penerjemahan dalam program ini.

Kebanyakan editor musik, dapat membuka dan menyimpan file MIDI dalam format 1. Data *delta-time* dalam file MIDI pada umumnya maksimal terdiri dari 2 byte, sehingga tidak lebih dari 16256 ketukan, atau dalam *variable-length quantity* dinyatakan sebagai FF 7F.

Dalam proses konversi berikutnya banyak sekali proses penyempurnaan dan perbaikan yang diakibatkan oleh kesalahan pemilihan metode yang digunakan dalam konversi. Memang dalam tugas akhir ini ada beberapa metode yang digunakan dalam proses, yaitu metode pengurutan data (*shorting*), metode pencarian jarak terpendek (*shortest path*), metode perbandingan (*branch and bound*), serta metode prioritas dalam penempatan jari. Penggabungan metode-metode tersebut timbul pada saat penyempurnaan program, sehingga terdapat beberapa metode yang sebelumnya tidak direncanakan untuk digunakan. Adapun metode-metode yang dipilih tersebut mungkin belum dapat memberikan hasil yang optimal, tetapi paling tidak sudah bisa memenuhi beberapa persyaratan dalam merepresentasikan notasi musik umum ke dalam notasi posisi jari pada fretboard.

BAB VI

PENUTUP

6.1. Kesimpulan

Dari hasil perancangan dan pembuatan perangkat lunak penerjemah notasi musik ke dalam bentuk tampilan posisi jari pada fret gitar ini, dapat ditarik beberapa kesimpulan sebagai berikut :

1. Perangkat lunak yang dibuat dapat membaca dan menerima masukan berupa file MIDI dengan format 0 yang terdiri atas sebuah *multi-channel track*, dan format 1 yang terdiri atas satu atau lebih *simultaneous track* yang berurutan, serta menggunakan sistem perhitungan waktu metrik (standar).
2. Perangkat lunak yang dibuat dapat menerima file MIDI yang berisi notasi musik yang bukan untuk alat musik gitar, tetapi dengan konsekuensi hasil penerjemahan menjadi tidak lengkap karena besar kemungkinan terdapat nada-nada yang diabaikan.
3. Jika data masukan berupa notasi musik untuk alat musik gitar, maka kemungkinan penerjemahan yang kurang tepat disebabkan oleh adanya notasi yang berpengaruh pada modifikasi suara atau efek suara.
4. Perangkat lunak ini dirancang hanya untuk menampilkan data hasil konversi ke layar monitor berupa posisi-posisi nada pada *fretboard* beserta nomor jari.

Perangkat lunak yang dibuat tidak terlepas dari kelemahan maupun kekurangan. Adapun kelemahan dan kekurangan tersebut antara lain :

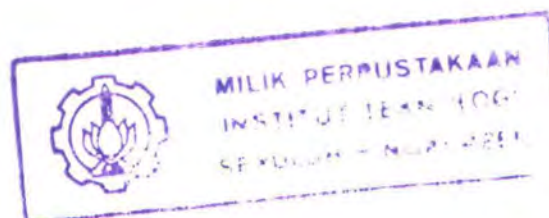
1. Kemungkinan terjadi kesalahan dalam proses pembacaan file MIDI, yaitu pada file yang mengandung data-data *real-time message*. Data tersebut adalah data yang digunakan untuk pengolah suara drum atau perkusi, yang bisa disisipkan pada event yang lain.
2. Proses konversi file notasi musik kurang efisien, karena setiap tahap konversi selalu disimpan dalam bentuk file sementara (*temporary*).
3. Metode yang digunakan dalam pencarian posisi maupun penentuan jari hanya mempertimbangkan bahwa suatu notasi musik dapat diterjemahkan ke dalam bentuk posisi-posisi jari pada *fretboard*, meskipun kadang sulit untuk merepresentasikannya pada permainan gitar karena faktor tempo dan irama suatu lagu maupun keterbatasan jari. Keadaan demikian kemungkinan disebabkan oleh notasi musik yang sebenarnya bukan untuk alat musik gitar.

6.2. Saran

Adapun berbagai saran yang dapat diberikan sehubungan dengan pengembangan perangkat lunak ini adalah :

1. Peningkatan kemampuan dalam mengkonversi file notasi musik, sehingga efek-efek suara yang ada dapat direpresentasikan dengan teknik serta pola perpindahan jari pada *fretboard*.
2. Proses konversi yang memanfaatkan fasilitas memori, yaitu berupa *array dinamis*, yang diharapkan dapat meminimalkan waktu proses.

3. Penggunaan metode lain yang tingkat optimasinya lebih baik dalam mengkonversi data notasi musik menjadi posisi jari pada *fretboard*, dan bisa memodifikasi suatu notasi musik yang bukan untuk alat musik gitar, sehingga mudah untuk merepresentasikannya pada permainan gitar.
4. Penambahan fasilitas yang bisa mengaktifkan perangkat multimedia, dimana bentuk hasil keluaran berupa tampilan visual bisa dilengkapi dengan suara nada yang dihasilkan untuk setiap posisi.



DAFTAR PUSTAKA

1. Francis A. Rozells & Clifford Cheam, **Rock & Heavy Metal Guitar**, Penerbit Muzikal, 1994
2. Teguh Wartono dkk., **Pengantar Pendidikan Seni Musik**, Penerbit Yayasan Kanisius, 1984
3. Neil Rowland, Jr., **"MIDI" Help File**, 1993
4. C. L. Liu, **Element of Discrete Mathematics**, second edition, McGraw-Hill Book Company, 1985
5. Ir. Tjutju Tarlih Dimyati, MSIE dan Ir. Ahmad Dimyati, MBA., **Operations Research**, Model-model Pengambilan Keputusan, Penerbit Sinar Baru Bandung, 1992
6. Patrick Henry Winston, **Artificial Intelligence**, second edition, Addison-Wesley Publishing Company, 1984
7. Andre LaMonthe, John Ratcliff, Mark Seminatore and Denise Tyler, **Tricks of the Game-Programming Gurus**, Sams Publishing, 1994
8. Andrew Wozniewicz with Namir Shammas, **Teach Yourself Borland Delphi in 21 Days**, Sams Publishing, 1995

LAMPIRAN

Program tugas akhir ini dirancang untuk mudah dioperasikan oleh pemakai, karena itu bentuk pilihan menu maupun tombol yang ada dibuat sangat sederhana, sesuai dengan fasilitas yang dibutuhkan untuk pemrosesan data yang ada dalam program ini.

Langkah-langkah menjalankan program ini adalah sebagai berikut :

1. Untuk membuka file yang akan diproses, pilih menu **File**, lalu pilih sub-menu **Open**, atau langsung tekan tombol *open file*. Selanjutnya akan tampil window *OpenFile*. Pilih nama file yang akan diproses.
2. Jika file yang dibuka bertipe *.MID, maka pada saat membuka file langsung dilanjutkan dengan proses konversi pertama, dimana data-data hasil konversi disimpan dalam file penampung GT1.TMP.
3. Langkah selanjutnya pilih menu **Process**, lalu pilih sub-menu **Compile**. Maka proses konversi akan dilanjutkan dengan penentuan posisi yang merupakan proses konversi kedua dengan data yang disimpan dalam file penampung GT2.TMP, dan penentuan jari sebagai proses konversi ketiga dengan data yang disimpan dalam file penampung GT3.TMP.
4. Untuk menampilkan data hasil konversi, pilih menu **Process**, lalu submenu **Run**, atau langsung tekan tombol *play*. Maka akan tampak notasi posisi jari pada fretboard, dan notasi nada yang dihasilkan.

5. Pada saat menampilkan data, pemakai bisa memperlambat atau mempercepat tempo perubahan data yang sedang ditampilkan dengan cara mengubah nilai dari *spin edit*, yang menampilkan nilai dari *division*. Atau jika akan menghentikan sementara posisi yang sedang ditampilkan, bisa dilakukan dengan menekan tombol *pause*.
6. Menutup file yang sedang eksis dilakukan dengan cara memilih menu **File** submenu **Close**, atau langsung tekan tombol *close*. Kemudian akan tampil pesan konfirmasi, apakah file yang akan ditutup tersebut akan disimpan sebagai file *.GTR (jika yang dibuka pada permulaan adalah file *.MID). Pesan ini juga muncul pada saat pemakai akan keluar dari program.
7. Untuk menyimpan data hasil keluaran maka pilih menu **File** submenu **Save**, atau langsung tekan tombol *save file*. Selanjutnya akan tampil window *SaveFile*.
8. Jika ingin keluar dari program, pilih menu **File** submenu **Exit**, atau langsung tekan tombol *exit*.